第 1 章　数字和逻辑基础

1.1　数字逻辑电路概述

1.1.1　模拟信号与数字信号

1．模拟信号

自然界中，人们可以感知的许多物理量，如速度、压力、温度、声音、质量及位置等，都有一个共同的特点：即它们在时间上和数值上是连续变化的信号。这种连续变化的物理量称为模拟量，表示模拟量的信号称为模拟信号。在工程实践中，通常用传感器将模拟量转换为与之成比例的电压或电流信号，再送到后续的信号处理系统（可能是模拟系统，也可能是数字系统，还可能是模拟与数字的混合系统）中做进一步处理。图 1-1 所示为两种电压的模拟信号的波形，模拟信号也可以是电流或其他物理量的波形。

2．数字信号

另一类物理量，它们在时间上和数值上都是离散的信号，或者说是不连续的，且其数值的大小和每次的变化都是某个值的整数倍，这种物理量称为数字量。表示数字量的信号称为数字信号。例如，普通指针式万用表在指示电压值时，通过表针的摆动和表面的刻度来指示电压值，其值是连续变化的；而数字式万用表则通过数字来指示电压值。例如，某一只数字式万用表只能够显示 3 位数字，它显示某电压值为 7.55V，比这个电压值再大一点的显示是 7.56V，但是该表无法显示 7.555V，它只能以每隔 0.01V 来分挡显示，说明它的指示值是不连续的，这种不能连续变化的显示信号就是数字信号。测出的电压值会存在一定误差，该误差取决于数字万用表的位数。

图 1-2 所示的波形是一种幅度取值只有两个值（0 和 1）的数字信号的波形，它是目前最常见的数字信号的波形，称为“二进制信号”。

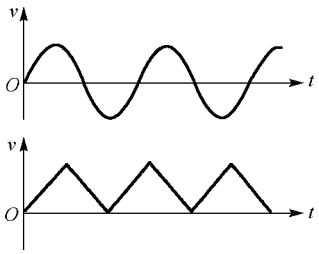


图 1-1　模拟信号波形

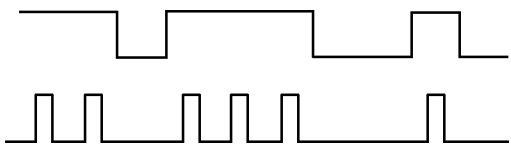


图 1-2　一种数字信号波形

1.1.2　模拟电路与数字电路

处理模拟信号的电子电路称为模拟电路，而工作于数字信号下的电子电路称为数字电路。所谓数字电路就是用于处理数字信号的电路。数字电路与模拟电路相比有很大的不同，数字电路主要是对数

字信号进行逻辑运算和数字处理，这些运算和处理有时是相当复杂的，这一点决定了对数字电路的识图不同于模拟电路的识图。

关于数字电路这里先介绍下列一些特点，以便对这种电路有初步印象。

（1）数字电路中只处理二进制中的“0”和“1”两种信号，“0”表示信号无，“1”表示信号有。从电路硬件这一角度上讲，电子电路中的元器件特别是三极管只工作在有信号和无信号两种状态，也就是数字电路中的三极管多半工作在开关状态，不像模拟电路中的三极管工作在放大状态。

（2）数字电路中，三极管的饱和状态与截止状态分别对应于数字信号中的“0”和“1”，可用三极管截止时输出的高电平表示数字信号的“1”状态，而用三极管饱和导通时输出的低电平表示数字信号中的“0”状态。三极管的这一工作状态与模拟电路是完全不同的，在进行数字电路识图时电路分析方法就不能与模拟电路中三极管放大状态的分析方法相同。

（3）由于数字信号只有“0”和“1”两种，那么对数字电路的工作要求就是能够可靠地区别信号为“0”和信号为“1”两种状态，因此对数字电路的精度要求不高，这适合于对数字电路进行集成化，加上对数字信号的处理和运算都是相当复杂的过程，所以数字电路中都是采用集成电路，且许多是大规模集成电路，这一点又使数字电路工作的分析增加了一份神秘的色彩。

（4）数字电路是实现逻辑功能和进行各种数字运算的电路。数字信号在时间上和数值上是不连续的，所以它在电路中只能表现为信号的有、无（或信号的高电平、低电平）两种状态。数字电路中用二进制数“0”和“1”来代表低电平和高电平两种状态，数字信号便可用“0”和“1”组成的代码序列来表示。因此，学习数字电路首先要了解有关二进制数知识，否则对数字电路的分析将寸步难行。

因此，概括起来，对于模拟电路和数字电路，其主要区别如下。

（1）工作任务不同

模拟电路研究的是输出信号与输入信号之间的大小、相位、失真等方面的关系；数字电路主要研究的是输出与输入间的逻辑关系（因果关系）。

（2）三极管的工作状态不同

模拟电路中的三极管工作在线性放大区，是一个放大元件；数字电路中的三极管工作在饱和或截止状态，起开关作用。所以，模拟电路和数字电路的基本单元电路、分析、设计的方法及研究的范围均不同。

1.1.3 数字信号参数

模拟信号的表示方式可以是数学表达式，也可以是波形图等。数字信号的表示方式可以是二值数字逻辑，以及由逻辑电平描述的数字波形。

1．理想数字信号的主要参数

数字信号是一种二值信号，用两个电平（高电平和低电平）分别来表示两个逻辑值（逻辑 1 和逻辑 0），如图 1-3 所示。

一个理想的周期性数字信号如图 1-4 所示，可以用下列参数来描述。

V_{m}：信号幅度，即高电平的值。

T：信号的重复周期，表示两个相邻脉冲之间的时间间隔；有时也使用频率 $f=1/T$ 表示单位时间内脉冲重复的次数。

t_{W}：脉冲宽度，即波形高电平维持的时间。

q（或β）：占空比，其定义为脉冲宽度 t_{W} 与信号周期 T 的比值，即

$$q(\%)=\frac{t_W}{T}\times 100\% \qquad (1\text{-}1)$$

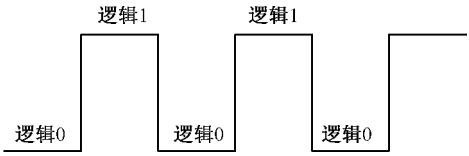


图 1-3　理想数字信号的波形

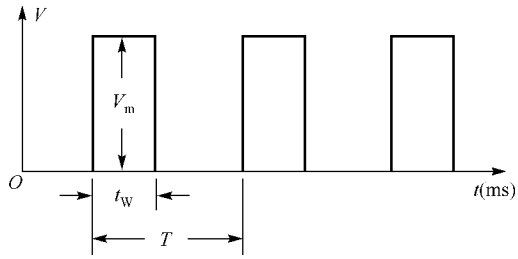


图 1-4　理想的周期性数字信号

2．实际数字信号波形及参数

在实际的数字系统中，数字信号并没有那么理想。当它从低电平跳变到高电平，或从高电平跳变到低电平时，边沿没有那么陡峭，而要经历一个过渡过程，分别用上升时间 t_r 和下降时间 t_f 描述，如图 1-5 所示。

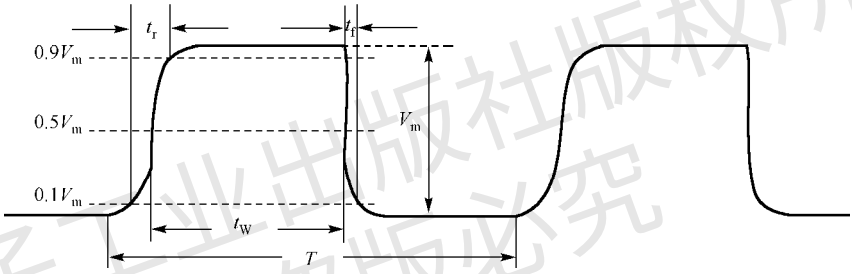


图 1-5　实际数字信号波形

该信号除了具有上述几个参数外，还定义了如下参数。

t_r：在脉冲的上升沿，从脉冲幅值的 10%上升到 90%所经历的时间（典型值 ns）；

t_f：在脉冲的下降沿，从脉冲幅值的 90%下降到 10%所经历的时间（典型值 ns）。

t_W：脉冲幅值从脉冲波形的上升沿上升到 $0.5V_m$ 开始，到下降沿下降至 $0.5V_m$ 为止的时间间隔。

1.1.4　数字电路的基本功能及其应用

数字电路的基本功能是对输入的数字信号进行算术运算和逻辑运算，即它具有一定的“逻辑思维”能力。数字电路是计算机、自动控制系统、各种智能仪表、现代通信系统等的基本电路，是学习这些专业知识的基础。

图 1-6 所示为一个典型的早期的电子系统的组成框图。框图中各部分电路功能如下。

（1）传感器

传感器是一种检测装置，能感受规定的被测量并按照一定的规律转换成可用信号的器件或装置，通常由敏感元件和转换元件组成，它是实现自动检测和自动控制的首要环节。该内容有专门的课程讲述。

（2）信号处理电路

信号处理电路完成根据实际需要对传感器送来的信号进行隔离、放大、滤波、运算、转换等处理。这部分电路一般为模拟电路，在模拟电子技术中论述。

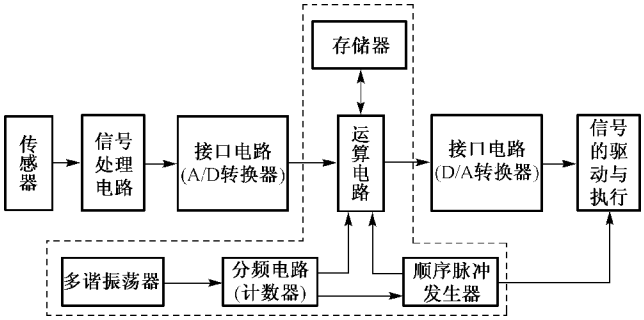


图 1-6 典型的电子系统的组成框图

（3）接口电路——A/D 转换器和 D/A 转换器

由于一般信号处理电路输出的是模拟信号，而数字运算电路只能处理数字信号，所以需通过 A/D 转换器把模拟信号转换成数字信号，运算电路对信号进行处理之后，再通过 D/A 转换器把数字信号转换成模拟信号。接口电路（即 A/D 转换器和 D/A 转换器）将在第 7 章进行介绍。

（4）运算电路

运算电路可完成信息的算术运算和逻辑运算，这部分内容在第 1 章后续部分和第 3 章中介绍。

（5）存储器

数字系统需要处理的数据量越来越大，速度也越来越快。有关存储器的知识及技术指标等在第 8 章中介绍。

访问存储器中指定地址的内容，需要对地址进行译码，即通过地址译码器寻址到相应的存储单元。译码器的内容将在第 3 章中介绍，对存储器中的内容如果需要显示，在该章也介绍了七段译码器及数码显示等内容。

（6）信号的驱动与执行

系统输出的控制信号一般功率有限，需通过功率放大后，驱动执行机构完成控制功能。功率放大电路的内容一般在模拟电路中介绍。

（7）波形的产生与整形电路——多谐振荡器

很多数字电路器件工作时都需要时钟信号，该信号可通过多谐振荡器获得，这部分的内容将在第 6 章中详细介绍。

（8）计数器（分频器）

不同数字电路器件需要的时钟频率可能有所不同，此时可通过分频电路——计数器获得不同频率的时钟信号。分频器的设计与分析将在第 5 章中介绍。

（9）顺序脉冲发生器

系统各部分的电路需要按照事先规定的时间顺序依次工作，顺序脉冲发生器便可完成此项工作，这部分的内容将在第 5 章中介绍。

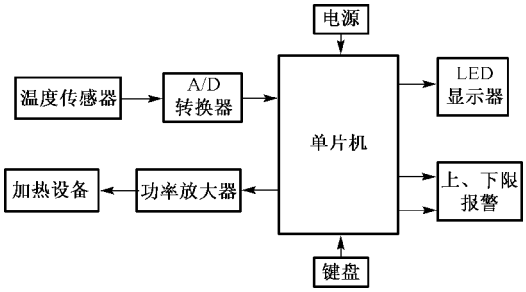


图 1-7 温度检测和控制电路实例

随着数字电子技术及计算机的迅猛发展，图 1-6 中虚线框内的运算电路、波形产生与整形电路、分频电路及顺序脉冲发生器等可采用计算机来完成，这种计算机可选用通用计算机，也可选用单片机、数字信号处理器（Digital Signal Processor，简称 DSP）及基于某总线的专用 CPU 等。图 1-7 所示为单片机的温度检测和控制电路实例。

1.2　数制、数制转换和算术运算简介

数制是数的表示方法，用数字表示数量大小时，经常要采用多位数码。我们把多位数码中每一位的构成方法以及从低位到高位的进位规则称为数制。

人们在日常生活和工作中习惯于使用十进制数，而在数字系统中，如计算机中，通常使用二进制数，有时也使用八进制数或十六进制数。

1.2.1　十进制数

在十进制数中，每一位有 0～9 共 10 个数码，所以计数的基数是 10。超过 9 的数必须用多位数表示，其中，低位和相邻高位之间的关系是“逢十进一”，故称为十进制。所以，十进制就是以 10 为基数的计数体制。

数码处于不同位置时，它所代表的数值是不同的，例如，十进制数 1234.56 可以表示为

$$1234.56 = 1\times10^3+2\times10^2+3\times10^1+4\times10^0+5\times10^{-1}+6\times10^{-2}$$

其中，10^3、10^2、10^1、10^0 分别为千位、百位、十位、个位数码的权，也就是相应位的 1 所代表的实际数值；10^{-1}、10^{-2} 是小数点以右各位数码的权值，是基数 10 的负幂。

根据以上分析，任意十进制数可表示为

$$(N)_{\mathrm{D}} = \sum_{i=-m}^{n} d_i \times 10^i \tag{1-2}$$

式中，d_i 是第 i 位的系数，它可以是 0～9 这 10 个数码中的任何一个数字；n、m 分别表示整数及小数部分的位数。

如果将式（1-2）中的 10 用基数 R 来代替，就可以得到任意进制数的表达式

$$(N)_{\mathrm{R}} = \sum_{i=-m}^{n} d_i \times R^i \tag{1-3}$$

式中，d_i 是第 i 位的系数，根据基数 R 的不同，它的取值为 R（0～R–1）个不同的数码。

构成数字电路的基本思路是把电路的状态与数码对应起来，十进制数的 10 个数码要求电路有 10 个完全不同的状态，这使得电路很复杂，因此用数字电路来存储或处理十进制数是不方便的。在数字电路中不直接处理十进制数，而常常采用二进制数、八进制数和十六进制数。

1.2.2　二进制数、八进制数和十六进制数

1．二进制数

在二进制数中，每一位仅有 0 和 1 两个数码，所以计数基数为 2。二进制数的进位规则是“逢二进一”，即 1+1=10（读为“壹零”）。注意：这里的“10”与十进制数的“10”是完全不同的，它并不代表数“拾”。左边的 1 表示 2^1 位数，右边的 0 表示 2^0 位数，即 $10 = 1\times2^1+0\times2^0$。因此，二进制数就是以 2 为基数的计数体制。

根据式（1-3），任何一个二进制数均可表示为

$$(N)_{\mathrm{B}} = \sum_{i=-m}^{n} d_i \times 2^i \tag{1-4}$$

式中，d_i 是第 i 位的系数，它可以是 0、1 两个数码中的任何一个数字，n、m 分别表示整数及小数部分的位数。式（1-4）也可以作为二进制数转换为十进制数的转换公式，例如

$$(1010.101)_{\mathrm{B}}=1\times2^3+0\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+0\times2^{-2}+1\times2^{-3}$$
$$=(10.625)_{\mathrm{D}}$$

式中，分别用下标 B（Binary）和 D（Decimal）表示括号中的数是二进制数和十进制数，有的书上也用 2 和 10 表示。

采用二进制数的主要优点是容易用器件实现。由于二进制的每一位只有 0、1 两种取值，因此可以用具有两个稳定状态的元件来表示一位二进制数。例如，用三极管的饱和与截止、继电器接点的闭合与断开、灯泡的亮与不亮等分别表示二进制数的 0 和 1。

采用二进制数的主要缺点是当数值较大时位数较多，使用起来不方便，也不习惯，因此在数字计算机的资料中常采用八进制数和十六进制数。

2．八进制数

在八进制数中，每一位有 0～7 共 8 个数码，计数的基数为 8。八进制数的进位规则是“逢八进一”。任意一个八进制数可以表示为

$$(N)_{\mathrm{O}}=\sum_{i=-m}^{n}d_i\times8^i \tag{1-5}$$

式中，d_i 是第 i 位的系数，它可以是 0～7 这 8 个数码中的任何一个数字。下标 O（Octal）用来表示八进制数，也可用 8 做下标。

式（1-5）也可以作为八进制数转换为十进制数的转换公式，例如

$$(123.4)_{\mathrm{O}}=1\times8^2+2\times8^1+3\times8^0+4\times8^{-1}$$
$$=(83.5)_{\mathrm{D}}$$

3．十六进制数

在十六进制数中，每一位有 16 个不同的数码，分别为 0～9、A、B、C、D、E、F，计数的基数为 16。十六进制数的进位规则为“逢十六进一”。任意一个十六进制数可以表示为

$$(N)_{\mathrm{H}}=\sum_{i=-m}^{n}d_i\times16^i \tag{1-6}$$

式中，d_i 是第 i 位的系数，它可以是 0～9、A、B、C、D、E、F 这 16 个数码中的任何一个。下标 H（Hexadecimal）用来表示十六进制数，也可用 16 做下标。

式（1-6）也可以作为十六进制数转换为十进制数的转换公式，例如

$$(2A5.6B)_{\mathrm{H}}=2\times16^2+\mathrm{A}\times16^1+5\times16^0+6\times16^{-1}+\mathrm{B}\times16^{-2}$$
$$=(752.41796875)_{\mathrm{D}}$$

为便于对照，将十进制、二进制、八进制及十六进制数之间的关系列于表 1-1 中。

表 1-1　几种数制之间的关系对照表

十进制数	二进制数	八进制数	十六进制数
0	00000	0	0
1	00001	1	1
2	00010	2	2

续表

十进制数	二进制数	八进制数	十六进制数
3	00011	3	3
4	00100	4	4
5	00101	5	5
6	00110	6	6
7	00111	7	7
8	01000	10	8
9	01001	11	9
10	01010	12	A
11	01011	13	B
12	01100	14	C
13	01101	15	D
14	01110	16	E
15	01111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13
20	10100	24	14

1.2.3　不同进制数间相互转换

1．二、八和十六进制数转换成十进制数

任意进制数（包括二进制数、八进制数和十六进制数）转换为十进制数的方法是，根据式（1-3）按权展开，然后将各项按十进制数相加，即得对应的十进制数，如前面 1.2.2 节所述。

2．十进制数转换成二、八和十六进制数

十进制数转换为二进制数时，要分成整数与小数两部分分别转换，然后将转换结果合成一个二进制数。

（1）整数转换

对于整数部分，假定十进制数为$(N)_{\mathrm{D}}$，等值的二进制数为$(d_{n-1}\cdots d_2d_1d_0)_{\mathrm{B}}$，则有

$$\begin{aligned}(N)_{\mathrm{D}} &= d_{n-1}2^{n-1}+\cdots+d_2 2^2+d_1 2^1+d_0 2^0\\ &= 2(d_{n-1}2^{n-2}+\cdots+d_2 2^1+d_1 2^0)+d_0\end{aligned} \tag{1-7}$$

若将式（1-7）中$(N)_{\mathrm{D}}$除以 2，则商为 $d_{n-1}2^{n-2}+\cdots+d_2 2^1+d_1 2^0$，余数为 d_0，因此把十进制数$(N)_{\mathrm{D}}$除以 2 后，余数即为对应二进制数的最低位 d_0，若将商记为 $(N)'_{\mathrm{D}}$，则

$$\begin{aligned}(N)'_{\mathrm{D}} &= d_{n-1}2^{n-2}+\cdots+d_2 2^1+d_1 2^0\\ &= 2(d_{n-1}2^{n-3}+\cdots+d_2 2^0)+d_1\end{aligned} \tag{1-8}$$

若再将式（1-8）中 $(N)'_{\mathrm{D}}$ 除以 2，则余数为 d_1，它对应二进制数的次低位 d_1。以此类推，每次以前次的商除以 2，即可求得二进制数的一位数字，连续除以 2 直到商为 0 为止，这时的余数为二进制数的最高位 d_{n-1}。从而由所有的余数求出二进制数。

【例 1-1】 将十进制数$(185)_{\mathrm{D}}$转换为二进制数。

根据上述原理，可将$(185)_{\mathrm{D}}$按如下的步骤转换为二进制数。

		余数	
2	185	1	……d_0
2	92	0	……d_1
2	46	0	……d_2
2	23	1	……d_3
2	11	1	……d_4
2	5	1	……d_5
2	2	0	……d_6
2	1	1	……d_7
	0		

故$(185)_D=(10111001)_B$。

当十进制数较大时，不必逐次除以 2，而是将十进制数和与其相当的 2 的乘幂项对比，使转换过程得到简化。

【例 1-2】 将$(135)_D$转换为二进制数。

解：由于 $2^7=128$，而 $135-128=7=2^2+2^1+2^0$，所以对应二进制数 $d_7=1$，$d_2=1$，$d_1=1$，$d_0=1$，其余各系数为 0，故得

$$(133)_D=(10000111)_B$$

值得指出，由于多数计算机或数字系统中只处理 4、8、16、32 或 64 位等的二进制数据，因此，转换得到的数据的位数需是规格化的位数，如转换结果为 10101，在纸上写为 10101 即可，但在计算机中就要高位补“0”，如果是 8 位计算机就要将它配成 8 位，则相应的高幂项应填以 0，其值不变，即

$$10101=00010101$$

（2）小数转换

设十进制小数为$(N)_D$，等值的二进制小数为$(0.d_{-1}d_{-2}d_{-3}\cdots d_{-m})_2$，则有

$$(N)_D=d_{-1}2^{-1}+d_{-2}2^{-2}+d_{-3}2^{-3}+\cdots+d_{-m}2^{-m} \tag{1-9}$$

将式（1-9）两边分别乘以 2，得

$$2(N)_D=d_{-1}+d_{-2}2^{-1}+d_{-3}2^{-2}+\cdots+d_{-m}2^{-(m-1)} \tag{1-10}$$

由此可见，将十进制小数乘以 2，所得乘积的整数部分为 d_{-1}，小数部分为 $d_{-2}2^{-1}+d_{-3}2^{-2}+\cdots+d_{-m}2^{-(m-1)}$。

同理，将乘积的小数部分再乘以 2，又可得到

$$2(d_{-2}2^{-1}+d_{-3}2^{-2}+\cdots+d_{-m}2^{-(m-1)})=d_{-2}+d_{-3}2^{-1}+\cdots+d_{-m}2^{-(m-2)} \tag{1-11}$$

也即乘积的整数部分就是 d_{-2}。

以此类推，将每次乘以 2 后所得乘积的小数部分再乘以 2，得到的整数即为对应位的二进制数，直到小数部分是 0 为止（或满足要求的精度为止），从而完成十进制小数转换成二进制小数。

【例 1-3】 将$(0.206)_D$转换为二进制数，要求其误差不大于 2^{-10}。

解：按上面介绍的方法计算，可得 d_{-1}、d_{-2}、…、d_{-9} 如下

	整数
$0.206\times2=0.412$	0……d_{-1}
$0.412\times2=0.824$	0……d_{-2}

$$0.824\times2=1.648 \qquad 1\cdots\cdots d_{-3}$$
$$0.648\times2=1.296 \qquad 1\cdots\cdots d_{-4}$$
$$0.296\times2=0.592 \qquad 0\cdots\cdots d_{-5}$$
$$0.592\times2=1.184 \qquad 1\cdots\cdots d_{-6}$$
$$0.184\times2=0.368 \qquad 0\cdots\cdots d_{-7}$$
$$0.368\times2=0.736 \qquad 0\cdots\cdots d_{-8}$$
$$0.736\times2=1.472 \qquad 1\cdots\cdots d_{-9}$$

由于最后的小数小于 0.5，根据“四舍五入”的原则，d_{-10} 应为 0，所以

$$(0.206)_D=(0.001101001)_B$$

综上所述，十进制数转换为二进制数采用的方法是除以 2 取余数（对于整数部分）和乘以 2 取整数（对于小数部分）的方法。十进制转换为八进制或十六进制的方法同上，只是基数 2 变成了 8 或 16。

3．二、八和十六进制数之间的转换

（1）二进制数与八进制数之间的转换

3 位二进制数共有 8 种组合，即 000、001、…、111，而一位八进制数有 8 个不同的数码，因此 3 位二进制数对应 1 位八进制数。

把二进制数转换为八进制数时，可先按 3 位进行分组，以小数点为基准，整数部分从右到左每 3 位为一组，最左边一组若不足 3 位的在高位补 0；小数部分从左到右每 3 位为一组，最右边一组若不足 3 位的在低位补 0。然后将每组二进制数用对应的八进制数表示，即得转换结果。

【例 1-4】 将二进制数$(1101111010.1011)_B$转换为八进制数。

解：$(1101111010.1011)_B=(001\ 101\ 111\ 010.101\ 100)_B$

$=(1572.54)_O$

将八进制数转换成二进制数时，只需将八进制数逐位用对应的 3 位二进制数表示，便得转换结果。

【例 1-5】 将八进制数$(567.32)_O$转换成二进制数。

解：$(567.32)_O=(101110111.01101)_B$

（2）二进制数与十六进制数之间的转换

由于 4 位二进制数对应 1 位十六进制数，因此把二进制数转换为十六进制数时，可先按 4 位进行分组，以小数点为基准，整数部分从右到左每 4 位为一组，最左边一组若不足 4 位的在高位补 0；小数部分从左到右每 4 位为一组，最右边一组若不足 4 位的在低位补 0。然后将每组二进制数用对应的十六进制数表示，即得转换结果。

【例 1-6】 将二进制数$(1101111010.1011)_B$转换为十六进制数。

解：$(1101111010.1011)_B=(0011\ 0111\ 1010.1011)_B$

$=(37A.B)_H$

将十六进制数转换成二进制数时，只需将十六进制数的每一位用对应的四位二进制数表示，便得转换结果。

1.2.4　符号数的表示方法

前面所述的二进制数，没有提及符号问题，是一种无符号数（总是正值）。那么数的正、负是如何表示的呢？通常采用的方法是原码表示法，即在二进制数的前面增加一位符号位。符号位为 0，表示这个数是正数，符号位为 1，表示这个数是负数。这种形式的数称为原码。

例如：

+89 → +1011001 → 01011001（原码）

–89 → –1011001 → 11011001（原码）

原码表示法的优点是简单，与真值转换方便。缺点是在用原码进行运算时需要进行加、减两种运算，用于运算的数字设备较复杂。为此引进了反码及补码表示形式。

正数的反码表示法同原码表示法。负数的反码表示法为：符号位为 1，数值位逐位取反。例如上面+89 和–89 的反码为

+89 → +1011001 → 01011001（原码） → 01011001（反码）

–89 → –1011001 → 11011001（原码） → 10100110（反码）

一般地，对于有效数字（不包括符号位）为 n 位的二进制数 N，它的反码$(N)_{反}$表示方法为

$$(N)_{反}=\begin{cases}N & （当N为正数）\\ (2^n-1)-N & （当N为负数）\end{cases} \tag{1-12}$$

即正数（当符号位为 0）的反码与原码相同，负数（当符号位为 1）的反码等于$(2^n-1)-N$。符号位保持不变。

在一些国外的教材中，也将式（1-12）定义的反码称为“1 的补码”（1’s Complement）。

正数的补码表示法同原码表示法。负数的补码表示法为：符号位为 1，数值位为反码加 1。例如上面+89 和–89 的补码为

+89 → +1011001 → 01011001 → 01011001 → 01011001（补码）

–89 → –1011001 → 11011001 → 10100110 → 10100111（补码）

一般地，对于有效数字（不包括符号位）为 n 位的二进制数 N，它的补码$(N)_{补}$表示方法为

$$(N)_{\mathrm{COMP}}=\begin{cases}N & （当N为正数）\\ 2^n-N & （当N为负数）\end{cases} \tag{1-13}$$

即正数（当符号位为 0）的补码与原码相同，负数（当符号位为 1）的补码等于2^n-N。符号位保持不变。

在一些国外的教材中，也将式（1-13）定义的补码称为“2 的补码”（2’s Complement）。

为了对照二进制数、原码、反码、补码的关系，表 1-2 列出了 8 位二进制数码对应的无符号二进制数、原码、反码、补码的值。

表 1-2 8 位二进制数码对应的二进制数、原码、反码、补码的值

8 位二进制数码	无符号二进制数	原码	反码	补码
00000000	0	+0	+0	+0
00000001	1	+1	+1	+1
00000010	2	+2	+2	+2
⋮	⋮	⋮	⋮	⋮
01111110	126	+126	+126	+126
01111111	127	+127	+127	+127
10000000	128	–0	–127	–128
10000001	129	–1	–126	–127
10000010	130	–2	–125	–126
⋮	⋮	⋮	⋮	⋮
11111110	254	–126	–1	–2
11111111	255	–127	–0	–1

1.2.5　多位二进制数的运算

在数字电路中，0 和 1 既可以表示逻辑状态，又可以表示数量的大小。当表示数量时，两个二进制数可以进行算术运算。二进制数的加、减、乘、除四种运算的运算规则与十进制数类似，两者唯一的区别在于进位或借位规则不同。

下面介绍无符号二进制数和有符号二进制数的算术运算。

1．无符号数的多位加法运算和减法运算

（1）加法运算

① 半加（本位加）概念

如果不考虑来自低位的进位而将两个 1 位二进制数相加，叫做半加。

无符号二进制数的半加规则是

$$0+0=0,\quad 0+1=1,\quad 1+0=1,\quad 1+1=\boxed{1}0$$

方框中的 1 是进位位，表示两个 1 相加“逢二进一”。可见，半加结果有两个输出：一是半加和；一是半加进位。

② 全加（带进位加）概念

实际作二进制加法时，一般地说，两个加数的位数都不会是 1 位，因而仅利用半加的概念是不能解决问题的。如果考虑来自低位的进位而将两个 1 位二进制数相加，叫做全加。

例如，两个 4 位二进制数 1011 和 1110 相加，则

$$\begin{array}{r} 1\ 0\ 1\ 1 \\ 1\ 1\ 1\ 0 \\ +\ 1\ 1\ 1\ 0 \quad \cdots\cdots\text{来自低位的进位} \\ \hline 1\ 1\ 0\ 0\ 1 \end{array}$$

由上述运算过程可以看到，右边的 1 位仅是两个加数相加，属于半加；而左边 3 位都是带进位的加法运算，即 2 个同位的加数和来自低位的进位三者相加，属于全加。

（2）减法运算

① 半减概念

如果不考虑来自低位的借位而将两个 1 位二进制数相减，叫做半减。

无符号二进制数的半减规则是

$$0-0=0,\quad 1-1=0,\quad 1-0=1,\quad 0-1=\boxed{1}1$$

方框中的 1 是借位位，表示 0 减 1 时不够减，向高位借 1。可见，半减结果有两个输出：一是半减差；一是半减借位。

② 全减概念

实际作二进制减法时，一般地说，被减数和减数的位数都不会是 1 位，因而仅利用半减的概念是不能解决问题的。如果考虑来自低位的借位而将两个 1 位二进制数相减，叫做全减。

例如，两个 4 位二进制数 1110 和 1011 相减，则

$$\begin{array}{r} 1\ 1\ 1\ 0 \\ 1\ 0\ 1\ 1 \\ -\ 0\ 1\ 1 \quad \cdots\cdots\text{来自低位的借位} \\ \hline 0\ 0\ 1\ 1 \end{array}$$

由上述运算过程可以看到，右边的 1 位仅是两个数相减，属于半减；而左边 3 位都是带借位的减法运算，即 2 个同位的数相减并考虑来自低位的借位，属于全减。

由于无符号二进制数中无法表示负数，因此要求被减数一定大于减数。

2．有符号数的多位加法运算和减法运算

有符号数的运算一般采用二进制补码的形式。

（1）加法运算

进行二进制补码的加法运算时，必须注意被加数补码与加数补码的位数相等，即让 2 个二进制补码的符号位对齐。

补码加法运算的规则：两个 n 位二进制数之和的补码等于该两数的补码之和，即

$$[X+Y]_{补}=[X]_{补}+[Y]_{补} \tag{1-14}$$

式（1-14）表明，当两个带符号数采用补码形式表示时，进行加法运算可以把符号位和数值位一起进行运算（若符号位有进位，则丢掉），结果为两数之和的补码形式。

【例 1-7】 已知 X = 33，Y = +15，Z = –15，求$[X+Y]_{补}$、$[X+Z]_{补}$。

解：

```
  +33    00100001          +33    00100001
  +15    00001111          –15    11110011
  +48    00110000          +18  1 00010010
                                 进位
                                 丢掉
```

故：$[X+Y]_{补}$=00110000　　$[X+Z]_{补}$=00010010

（2）减法运算

补码减法运算的规则：两个 n 位二进制数之差的补码等于被减数的补码与减数取负的补码之和，即

$$[X-Y]_{补}=[X]_{补}+[-Y]_{补} \tag{1-15}$$

减数取负的补码即已知$[Y]_{补}$求$[-Y]_{补}$的过程，称为变补或求负。变补或求负的规则是对$[Y]_{补}$的每一位（包括符号位）都按位取反，然后再加 1，结果就是$[-Y]_{补}$。例如

若$[15]_{补}$=00001111

则$[-15]_{补}$=11110001

【例 1-8】 已知 X=33，Y=+15，求$[X-Y]_{补}$。

解：$[X]_{补}$=00100001B　$[Y]_{补}$=00001111B　$[-Y]_{补}$=11110001B

$[X-Y]_{补}=[X]_{补}+[-Y]_{补}$

```
    00100001
+   11110001
  1 00010010
  进位
  丢掉
```

则$[X-Y]_{补}$=00010010。

【例 1-9】 已知 X = 33，Y = –15，求$[X-Y]_{补}$。

解：$[X]_{补}$=00100001B　$[Y]_{补}$=11110001B　$[-Y]_{补}$=00001111B

$[X-Y]_{补}=[X]_{补}+[-Y]_{补}$

```
    00100001
+   00001111
    00110000
```

则$[X-Y]_{补}$=00110000。

通过变补相加法使减法运算变为加法运算，基于这一点，数字系统及计算机中常采用补码表示带符号数。

两个带符号数运算时，如果运算结果大于数字设备所能表示的范围，就产生溢出，这时运算结果发生错误。如果某数字设备采用 8 位二进制数码，则它所能表示补码数的范围是 01111111～10000000（如表 1-2 所示），即+127～–128。如果运算结果大于+127 或小于–128 均产生溢出。

两个符号相反的数相加不会产生溢出，但两个符号相同的数相加，有可能产生溢出。

3．乘法运算简介

【例 1-10】 计算两个二进制数 1010 和 0101 的积。

解：

$$\begin{array}{r} 1010 \\ \times\ 0101 \\ \hline 1010 \\ 0000 \\ 1010 \\ 0000 \\ \hline 110010 \end{array}$$

所以 1010×0101=110010。

由上述运算过程可见，乘法运算是由左移被乘数与加法运算组成的。

4．除法运算简介

【例 1-11】 计算两个二进制数 1010 和 111 的商。

解：

$$\begin{array}{r} 1.011\cdots \\ 111\overline{)\,1010} \\ 111 \\ \hline 1100 \\ 111 \\ \hline 1010 \\ 111 \\ \hline 11 \end{array}$$

所以 1010÷111=1.011…。

由上述运算过程可见，除法运算是由右移被除数与减法运算组成的。

1.3　常 用 码 制

计算机等数字系统能直接处理的是二进制数码，为便于对数值、文字、符号、图形、声音和图像等信息进行处理，常将它们按照一定规则用多位二进制数码来表示，这种表示过程称为编码，而把这种表示特定信息的多位二进制数码称为代码，形成代码的规则，称为码制。

1.3.1　数字编码

1．自然二进制数的编码

自然二进制数的编码前面已经介绍过了，n 位自然二进制数的编码为 $d_{n-1}\cdots d_2d_1d_0$，如 4 位自然二进制数的编码为 0000（0）、0001（1）、…、1111（15）。

2．带符号二进制数的编码

带符号二进制数的编码就是在自然二进制数的编码前加上符号位，如为正数，符号位为 0，如为负数，符号位为 1。

3．BCD 码（Binary Coded Decimal）

在数字电路中，常用二-十进制码，也叫做 BCD（Binary Coded Decimal）码，所谓 BCD 码，就是用 4 位二进制数来表示 1 位十进制数。4 位二进制数有 16 种不同的组合方式，即 16 种代码，而 1 位十进制数只有 10 个数码，因而，从 16 种组合中选用其中 10 种组合来进行编码时，将会有不同的编码方案。表 1-3 所示为几种常用的 BCD 码。

8421 码是最常用的一种 BCD 码。它是由 4 位自然二进制数 16 种组合的前 10 种组成的，即 0000～1001，其余 6 种组合是无效的。其编码中每位的权都是固定数，称为位权。从高位到低位的权值分别为 8、4、2、1，这也是 8421 码得名的由来，它属于有权码。若要表示十进制数 5678，可用 8421 码表示为 0101 0110 0111 1000，即

$$(5678)_{\mathrm{D}}=(0101\ 0110\ 0111\ 1000)_{8421\mathrm{BCD}}$$

表 1-3 中的有权码还有 2421A 码、2421B 码、5421 码，其代码的每一位都具有一固定的权值的编码。一般情况下，有权码的十进制数与二进制数之间可用式（1-16）来表示

$$(N)_{\mathrm{D}}=W_3d_3+W_2d_2+W_1d_1+W_0d_0 \tag{1-16}$$

式中，W_3～W_0 为二进制码中各位的权。

表 1-3 中的余 3 码、余 3 循环码和格雷码均属于无权码，无权码代码的每一位没有固定的权值。

4．余三码

余 3 码是在每组 8421 码上加 0011 形成的，若把余 3 码的每组代码看成 4 位二进制数，那么每组代码均比相应的十进制数多 3，故称为余 3 码。

余 3 码的特点是具有相邻性，任意两个相邻代码之间仅有一位取值不同，例如 4 和 5 两个代码 0100 和 1100 仅 d_3 不同。余 3 码可以看成是将格雷码首尾各 3 种状态去掉而得到的（如表 1-3 所示）。

表 1-3　几种常用的 BCD 码

编码种类 / 十进制数	8421 码	2421A 码	2421B 码	5421 码	余 3 码	余 3 循环码	BCD 格雷码
0	0000	0000	0000	0000	0011	0010	0000
1	0001	0001	0001	0001	0100	0110	0001
2	0010	0010	0010	0010	0101	0111	0011
3	0011	0011	0011	0011	0110	0101	0010
4	0100	0100	0100	0100	0111	0100	0110
5	0101	0101	1011	1000	1000	1100	0111
6	0110	0110	1100	1001	1001	1101	0101
7	0111	0111	1101	1010	1010	1111	0100
8	1000	1110	1110	1011	1011	1110	1100
9	1001	1111	1111	1100	1100	1010	1101
权	8421	2421	2421	5421	无	无	无

1.3.2　可靠性编码

可靠性编码的作用是为了提高系统的可靠性。代码在形成和传送过程中都可能发生错误，为了使代码本身具有某种特征或能力，尽可能减少错误的发生，或者出错后容易被发现，甚至查出错误的码位后能予以纠正，因而形成了各种编码方法。下面介绍两种常用的可靠性编码。

1．格雷码（循环码、反射码）

格雷码（Gray Code）又称为循环码。格雷码的构成方法是每一位的状态变化都按一定的顺序循环。以 4 位格雷码为例，如表 1-4 所示。如果从 0000 开始，最右边一位的状态按 0110 顺序循环变化；右边第二位的状态按 00111100 顺序循环变化；右边第三位按 0000111111110000 顺序循环变化。自右向

左，每一位状态循环中连续的 0、1 数目增加一倍。由于 4 位格雷码只有 16 个，所以最左边一位的状态只有半个循环，即 0000000011111111。

格雷码的基本特点如下。

（1）具有相邻性，即两个相邻代码之间仅有 1 位数码不同。当数字电路中采用格雷码计数时，由于相邻代码只有一位数码发生变化，因此，可减少电路中竞争与冒险的可能性。

（2）具有反射特性。观察格雷码可以看出，以中轴为对称的两组码最高位相反，其余位相同。因此称格雷码具有反射特性，格雷码也称为反射码。

表 1-4 是 4 位格雷码，同理两位格雷码共 4 个代码为 00、01、11、10；3 位格雷码共 8 个代码为 000、001、011、010、110、111、101、100，均满足上述格雷码的特点。

另外十进制数 0～9 的格雷码即 BCD 格雷码，如表 1-3 所示。

2．奇偶校验码

二进制代码在传送过程中，常会由于干扰而发生错误，即有的 1 错成了 0，或有的 0 错成了 1。奇偶校验码是用来检验这种错误的代码。它由信息位和校验位两部分组成，信息位就是需要传送的信息本身，可由任何一种二进制码组成，位数不限；奇偶校验位仅有 1 位，可以放在信息位的前面，也可以放在后面，它使整个代码中 1 的个数按照预先规定成为奇数或偶数。

当采用奇校验时，信息位和校验位中 1 的总个数为奇数；当采用偶校验时，信息位和校验位中 1 的总个数为偶数。因此，校验位的代码可能为 1，也可能为 0，具体要根据选择校验方式（奇校验或偶检验）以及信息位中 1 的个数来确定。表 1-5 所示为 8421BCD 码的奇偶校验码，给出了由 4 位信息位和 1 位奇偶校验位共 5 位数码构成的奇偶校验码。

表 1-4　4 位格雷码与二进制代码

编码顺序	二进制代码	格雷码
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

表 1-5　8421BCD 码的奇偶校验码

十进制数	信息位	校验位	信息位	校验位
	8421BCD	奇校验	8421BCD	偶校验
0	0000	1	0000	0
1	0001	0	0001	1
2	0010	0	0010	1
3	0011	1	0011	0
4	0100	0	0100	1
5	0101	1	0101	0
6	0110	1	0110	0
7	0111	0	0111	1
8	1000	0	1000	1
9	1001	1	1001	0

当数据发送端发送出奇偶校验码，接收端接收到数据后，当发现信息位和校验位中 1 的总个数不正确时，就认为接收到的是错误代码。例如，在表 1-5 中传送信息使用偶校验码时，如果收到信息的每一组代码中 1 的总个数是偶数，认为接收到的数据是正确的。若收到的代码为如 00001、00111 等，则由于 1 的总个数为奇数，就是错误代码，也称为非法码，说明传送的数据发生了错误。

奇偶校验只能检测出一位或奇数位错码，但无法测定哪一位出错，也不能自行纠错。若两位或偶数位同时出现错误，则奇偶校验码无法检测出错误，但这种出错概率极小，且奇偶校验码容易实现，故被广泛应用。

3．更复杂的可靠性编码方法

汉明（Hamming）码等编码不但可以查出错误，还能校正错误，读者若需要进一步了解，请参阅其他有关资料。

1.3.3 信息交换代码

在计算机等数字系统中，处理的不仅有数字，还有字母、标点、运算符号及其控制符号等，它们也需要用二进制代码来表示，称为字符代码。目前使用最广泛的是由美国国家标准化协会（ANSI）制定的一种信息代码 ASCII 码——American Standard Code for Information Interchange（美国标准信息交换码）。ASCII 码已经由国际标准化组织（ISO）认定为国际通用的标准代码。

ASCII 码是一组 7 位二进制代码（$b_6b_5b_4b_3b_2b_1b_0$），共 128 个，如表 1-6 所示。其中，包括 0～9 这 10 个数字代码，大、小写英文字母 52 个代码，32 个各种符号的代码以及 34 个控制码。34 个控制码的含义列于表 1-7 中。

请读者自己结合表 1-6 分析总结 10 个数字 0～9、26 个大写字母 A～Z 和 26 个小写字母 a～z 的 ASCII 码的十六进制值，有什么特点。

表 1-6 ASCII 编码表

$b_3b_2b_1b_0$	$b_6b_5b_4$							
	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	“	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	‘	7	G	W	g	w
1000	BS	CAN	(	8	H	S	h	s
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	’	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	～
1111	SI	US	/	?	O	_	o	DEL

表 1-7 ASCII 码中控制码的含义

代码	含义	
NUL	Null	空白，无效
SOH	Start of heading	标题开始
STX	Start of text	正文开始
ETX	End of text	文本结束
EOT	End of transmission	传输结束
ENQ	Enquiry	询问
ACK	Acknowledge	承认
BEL	Bell	报警
BS	Backspace	退格
HT	Horizontal tab	横向制表
LF	Line feed	换行
VT	Vertical tab	垂直制表
FF	Form feed	换页
CR	Carriage return	回车
SO	Shift out	移出
SI	Shift in	移入
DLE	Date link escape	数据通信换码
DC1	Device control 1	设备控制 1
DC2	Device control 2	设备控制 2
DC3	Device control 3	设备控制 3
DC4	Device control 4	设备控制 4
NAK	Negative acknowledge	否定
SYN	Synchronous idle	空转同步
ETB	End of transmission block	信息块传输结束
CAN	Cancel	作废
EM	End of medium	媒体用毕
SUB	Substitute	代替，置换
ESC	Escape	扩展
FS	File separator	文件分隔
GS	Group separator	组分隔
RS	Record separator	记录分隔
US	Unit separator	单元分隔
SP	Space	空格
DEL	Delete	删除

1.4　逻辑代数基础

逻辑代数是一种描述客观事物逻辑关系的数学方法。它是英国数学家乔治·布尔（George Boole）在 1847 年首先提出来的，所以又称为布尔代数（或开关代数）。

逻辑代数用于研究二值逻辑变量的逻辑运算规律，所以也称为二值逻辑代数。在普通代数中已经知道，变量的取值可以从–∞～+∞，而在逻辑代数中，变量的取值只能是 0 和 1，而且必须注意逻辑代数中的 0 和 1 与十进制数中的 0 和 1 有着完全不同的含义，它代表了矛盾和对立的两个方面，如开关的闭合与断开、灯泡的亮与灭、电平的高和低、一件事情的是与非和真与假等。

如果以 *n* 个逻辑变量作为输入，形成新的逻辑变量的运算，称为逻辑运算。以运算结果作为输出，那么当输入变量的取值确定之后，输出的取值便随之而定。

1.4.1　基本逻辑运算和复合逻辑运算

逻辑代数的基本逻辑运算包括与、或、非 3 种运算。下面结合指示灯控制电路的实例分别讨论。

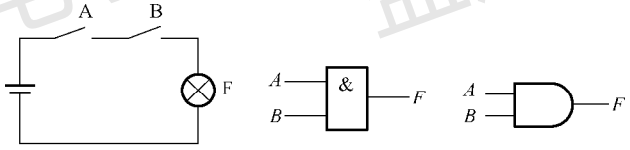
1．基本逻辑运算

（1）与运算

图 1-8(a)所示为两个开关串联控制指示灯的电路。由图可知，只有 A 与 B 两个开关同时闭合时，指示灯 F 才会亮；如果有一个开关断开或两个开关均断开，则指示灯不亮。

由此得到这样的逻辑关系：只有当一件事的几个条件全部具备之后，这件事才发生。这种关系称为与逻辑，也叫做逻辑与。

若以 *A*、*B* 表示开关的状态，并以 1 表示开关闭合，以 0 表示开关断开；以 *F* 表示灯的状态，用 1 表示灯亮，用 0 表示灯不亮，可得出对输入开关 *A*、*B* 所有取值的组合与其所对应的指示灯 *F* 的状态所构成的表格，称为真值表，如表 1-8 所示。



(a) 电路图　(b) 矩形符号　(c) 特异性符号

图 1-8　与运算

表 1-8　与逻辑真值表

A	*B*	*F*
0	0	0
0	1	0
1	0	0
1	1	1

在逻辑代数中，上述逻辑关系可用逻辑表达式来描述，可写为

$$F = A \cdot B \tag{1-17}$$

式中，小圆点“·”表示 *A*、*B* 的与运算，也称为逻辑乘。在不致引起混淆的前提下，乘号“·”可省略。在有些文献中，也有用符号∧、∩表示与运算。与运算的规则为

$$0 \cdot 0 = 0 \qquad 0 \cdot 1 = 0 \qquad 1 \cdot 0 = 0 \qquad 1 \cdot 1 = 1$$

在数字电路中，实现逻辑与运算的单元电路叫做与门，与门的逻辑符号如图 1-8(b)、图 1-8(c)所示。其中，图 1-8(b)的符号为国标（GB 4728.12—1996）符号，即矩形符号，国内教材常用；图 1-8(c)的符号为 IEEE（电气与电子工程师协会）标准的特定外形符号，即特异性符号，目前在国外教材和 EDA 软件中普遍使用。

（2）或运算

图 1-9(a)所示为两个开关并联控制指示灯的电路。由图可见，只要任何一个开关（A 或 B）闭合或两个都闭合，指示灯 F 都会亮；如果两个开关都断开，则指示灯不亮。

由此得到另一种逻辑关系：当一件事情的几个条件中只要有一个条件得到满足，这件事就会发生。这种关系称为或逻辑，也叫做逻辑或。或逻辑的真值表如表 1-9 所示。

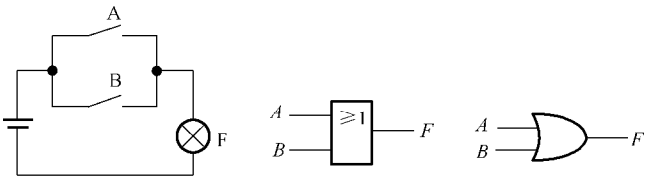
在逻辑代数中，上述逻辑关系可用逻辑表达式来描述，可写为

$$F = A + B \tag{1-18}$$

式中，符号“+”表示 A、B 的或运算，也称为逻辑或。在有些文献中，也有用符号∨、∪表示或运算。或运算的规则为

$$0+0=0 \qquad 0+1=1 \qquad 1+0=1 \qquad 1+1=1$$

在数字电路中，实现或逻辑运算的单元电路叫做或门，或门的逻辑符号如图 1-9(b)、图 1-9(c)所示。其中，图 1-9(b)所示的符号为矩形符号，图 1-9(c)所示的符号为特异性符号。



(a) 电路图　(b) 矩形符号　(c) 特异性符号

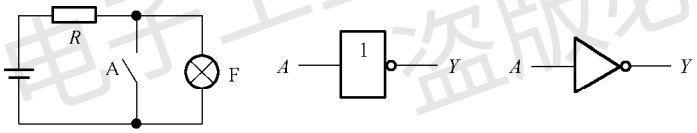
图 1-9　或运算

表 1-9　或逻辑真值表

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

（3）非运算

由图 1-10(a)所示的电路可知：当开关 A 闭合时，指示灯不亮；而当开关 A 断开时，指示灯亮。它所反映的逻辑关系是：当条件不具备时，事情才会发生。这种关系称为逻辑非，也叫做非逻辑。非逻辑的真值表如表 1-10 所示。



(a) 电路图　(b) 矩形符号　(c) 特异性符号

图 1-10　非运算

表 1-10　非逻辑真值表

A	F
0	1
1	0

在逻辑代数中，上述逻辑关系可用逻辑表达式来描述，可写为

$$F = \overline{A} \tag{1-19}$$

式中，A 上的短划线“–”表示非运算，也称为逻辑非。在有些文献中，也有用符号“～”、“¬”、“′”表示非运算。非运算的规则为：$\overline{0}=1$ 或 $\overline{1}=0$。

在数字电路中，实现逻辑非运算的单元电路叫做非门，非门的逻辑符号如图1-10(b)、图1-10(c)所示。其中，图 1-10(b)所示的符号为矩形符号，图 1-10(c)所示的符号为特异性符号。

2．复合逻辑运算

实际的逻辑问题往往比与、或、非复杂得多，不过它们都可以用与、或、非这 3 种基本逻辑运算组合而成。最常见的复合逻辑运算有与非、或非、异或、同或等。

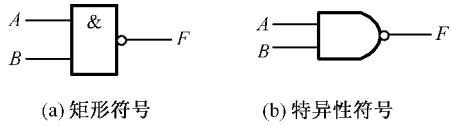
（1）与非

与非是由与运算和非运算组合而成的。其真值表如表 1-11 所示，逻辑符号如图 1-11 所示。逻辑表达式可写成

$$F=\overline{A\cdot B} \tag{1-20}$$

表 1-11　与非逻辑真值表

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0



(a) 矩形符号　　(b) 特异性符号

图 1-11　与非逻辑符号

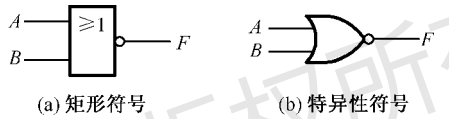
（2）或非

或非是由或运算和非运算组合而成的。其真值表如表 1-12 所示，逻辑符号如图 1-12 所示。逻辑表达式可写成

$$F=\overline{A+B} \tag{1-21}$$

表 1-12　或非逻辑真值表

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0



(a) 矩形符号　　(b) 特异性符号

图 1-12　或非逻辑符号

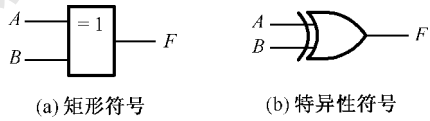
（3）异或

异或的逻辑关系是：当两个输入信号相同时，输出为 0；当两个输入信号不同时，输出为 1。其真值表如表 1-13 所示，逻辑符号如图 1-13 所示。逻辑表达式可写成

$$F=A\overline{B}+\overline{A}B=A\oplus B \tag{1-22}$$

表 1-13　异或逻辑真值表

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0



(a) 矩形符号　　(b) 特异性符号

图 1-13　异或逻辑符号

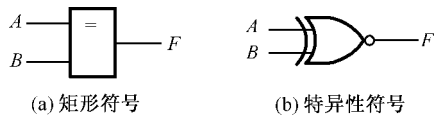
（4）同或

同或和异或的逻辑关系刚好相反：当两个输入信号相同时，输出为 1；当两个输入信号不同时，输出为 0。其真值表如表 1-14 所示，逻辑符号如图 1-14 所示。逻辑表达式可写成

$$F=AB+\overline{A}\ \overline{B}=A\odot B \tag{1-23}$$

表 1-14　同或逻辑真值表

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1



(a) 矩形符号　　(b) 特异性符号

图 1-14　同或逻辑符号

由上可知：同或和异或互为反运算，即

$$\overline{A\oplus B}=A\odot B \qquad A\odot B=\overline{A\oplus B} \tag{1-24}$$

1.4.2 基本公式和常用公式

1. 基本公式

根据上面介绍的逻辑与、或、非 3 种基本运算规则，可以推导出逻辑运算的基本公式，如表 1-15 所示。

表 1-15 逻辑运算的基本公式

类别	名称	逻辑与	逻辑或
变量与常量的关系	0-1 律	$A\cdot 0=0$ $A\cdot 1=A$	$A+0=A$ $A+1=1$
和普通代数相似的定律	交换律	$A\cdot B=B\cdot A$	$A+B=B+A$
	结合律	$A\cdot(B\cdot C)=(A\cdot B)\cdot C$	$A+(B+C)=(A+B)+C$
	分配律	$A\cdot(B+C)=A\cdot B+A\cdot C$	$A+(B\cdot C)=(A+B)\cdot(A+C)$
逻辑代数特殊的定律	互补律	$A\cdot\overline{A}=0$	$A+\overline{A}=1$
	重叠律	$A\cdot A=A$	$A+A=A$
	反演律（摩根定律）	$\overline{A\cdot B}=\overline{A}+\overline{B}$	$\overline{A+B}=\overline{A}\cdot\overline{B}$
	还原律	$\overline{\overline{A}}=A$	

表 1-15 所列公式均可通过列逻辑真值表证明其正确性。例如，要证明重叠律 $A+A=A$ 时，令 $A=0$，则 $A+A=0+0=0=A$；再令 $A=1$，则 $A+A=1+1=1=A$，可见，$A+A=A$。

在以上所有基本公式中，反演律具有特殊重要的意义，它又称为摩根定律，经常用于求一个原函数的非函数或者对逻辑函数进行变换。

【例 1-12】 用真值表证明反演律公式 $\overline{A\cdot B}=\overline{A}+\overline{B}$ 和 $\overline{A+B}=\overline{A}\cdot\overline{B}$ 的正确性。

解：将 A、B 的各种取值组合代入等式的两边，算出结果填入真值表 1-16 中。可见公式 $\overline{A\cdot B}=\overline{A}+\overline{B}$ 和 $\overline{A+B}=\overline{A}\cdot\overline{B}$ 等式两边的真值表对应相同，所以等式成立，公式正确。

2. 常用公式

利用表 1-15 所示的基本公式，可以推出其他一些公式，如表 1-17 所示。它们在逻辑函数的公式法化简中经常用到。

表 1-16 例 1-12 的真值表

A	B	$\overline{A\cdot B}$	$\overline{A}+\overline{B}$	$\overline{A+B}$	$\overline{A}\cdot\overline{B}$
0	0	1	1	1	1
0	1	1	1	0	0
1	0	1	1	0	0
1	1	0	0	0	0

表 1-17 一些常用公式

吸收律	$A+AB=A$	$A(A+B)=A$
	$A+\overline{A}B=A+B$	
	$AB+\overline{A}C+BC=AB+\overline{A}C$	
结合律	$AB+A\overline{B}=A$	$(A+B)(A+\overline{B})=A$

利用表 1-15 所列的基本公式，各式分别证明如下。

（1）$A+AB=A$

证明： $A+AB=A\cdot 1+AB=A(1+B)=A\cdot 1=A$

该式说明，在两个乘积项相加时，若其中一项是另一乘积项的因子，则该乘积项是多余的，可以去掉。

（2）$A(A+B)=A$

证明： $A(A+B)=AA+AB=A+AB=A$

该式说明，变量 A 和包含 A 的和相乘时，其结果等于 A，即可以将和消掉。

（3） $A+\overline{A}B=A+B$

证明： $A+\overline{A}B=(A+\overline{A})\cdot(A+B)=1\cdot(A+B)=A+B$

这说明一个与或表达式中，如果一项的反是另一个乘积项的因子，则该因子是多余的，可以消去。

（4） $AB+\overline{A}C+BC=AB+\overline{A}C$

证明： $AB+\overline{A}C+BC=AB+\overline{A}C+BC(A+\overline{A})$

$$=AB+\overline{A}C+ABC+\overline{A}BC$$
$$=AB(1+C)+\overline{A}C(1+B)$$
$$=AB+\overline{A}C$$

该式说明，如果与或表达式中，两个乘积项分别包含同一因子的原变量和反变量，而两项的剩余因子正好组成第 3 项，则第 3 项是多余的，可以去掉。

推广：如果第 3 项是包含剩余因子的乘积项，公式依然成立，即

$$AB+\overline{A}C+BCD=AB+\overline{A}C$$

（5） $AB+A\overline{B}=A$

证明： $AB+A\overline{B}=A(B+\overline{B})=A\cdot1=A$

可见，若两个乘积项中分别包含同一因子的原变量和反变量，而其他因子相同时，则两个乘积项相加可以合并成一项，并消去互为反变量的因子。

（6） $(A+B)(A+\overline{B})=A$

证明： $(A+B)(A+\overline{B})=A\cdot A+A\cdot\overline{B}+B\cdot A+B\cdot\overline{B}$

$$=A+A\overline{B}+AB$$
$$=A$$

可见，若两个和项中分别包含同一因子的原变量和反变量，而和项的另一因子相同时，则两个和项相乘后结果为相同的那个因子。

1.4.3　基本规则

逻辑代数中有 3 个重要规则：代入规则、反演规则和对偶规则。运用这些规则可将原有的公式加以扩展，从而推出一些新的运算公式。

1．代入规则

在任何一个逻辑等式中，若将等式两边所出现的同一变量代之以另一函数式，则等式仍然成立，这一规则称为代入规则。

因为任何逻辑函数式和被代替的变量一样，只有 0 和 1 两种状态，所以代入后等式依然成立。利用代入规则可以把表 1-15 的基本公式和表 1-17 的常用公式推广为多变量的形式和证明恒等式。

【例 1-13】 用代入规则证明：摩根定律也适用于多变量的情况。

解： 已知两变量的摩根定律为

$$\overline{A+B}=\overline{A}\cdot\overline{B}\qquad\qquad\overline{A\cdot B}=\overline{A}+\overline{B}$$

现以(*B*+*C*)代入左边等式中 *B* 的位置，以 *B*·*C* 代入右边等式中 *B* 的位置，于是得到

$$\overline{A+(B+C)}=\overline{A}\cdot\overline{B+C}=\overline{A}\cdot\overline{B}\cdot\overline{C}$$
$$\overline{A\cdot(B\cdot C)}=\overline{A}+\overline{B\cdot C}=\overline{A}+\overline{B}+\overline{C}$$

即

$$\overline{A+B+C}=\overline{A}\cdot\overline{B}\cdot\overline{C}$$

$$\overline{A\cdot B\cdot C}=\overline{A}+\overline{B}+\overline{C}$$

2．反演规则

对于任意一个逻辑函数式 F，若将式中所有的“·”换成“+”，“+”换成“·”，0 换成 1，1 换成 0，原变量换成反变量，反变量换成原变量，则得到的结果就是 $\overline{F}$。这一规则称为反演规则。

摩根定律就是反演规则的一个特例，所以摩根定律又称为反演律。利用反演规则可以十分方便地求出已知函数的反函数。

在使用反演律时，还需注意遵守以下两个规则：

（1）仍需遵守“先括号、然后乘、最后加”的运算优先次序；

（2）不属于单个变量上的反号应保留不变。

【例 1-14】 求函数 $F=A(B+C)+\overline{C}D$ 的反函数 $\overline{F}$。

解：根据反演规则可写出

$$\overline{F}=(\overline{A}+\overline{B}\cdot\overline{C})\cdot(C+\overline{D})$$

【例 1-15】 已知 $F=A+\overline{B\overline{C}+\overline{D+\overline{E}}}$，求 $\overline{F}$。

解：按照反演规则，并保留反变量以外的非号不变，得

$$\overline{F}=\overline{A}\cdot\overline{(\overline{B}+C)\cdot\overline{\overline{D}\cdot E}}$$

3．对偶规则

如果两个逻辑式相等，则它们的对偶式也相等，这就是对偶规则。

所谓对偶式是这样定义的：对于任何一个逻辑式 F，若把 F 中所有的“·”换成“+”，“+”换成“·”，0 换成 1，1 换成 0，并保持原来的运算顺序，则得到一个新的逻辑式 F'，那么 F 和 F' 互为对偶式。

利用对偶式可以证明恒等式。表 1-15 中第 3 列和第 4 列的公式是互为对偶关系的。

例如，若 $F=A(B+C)$，则 $F'=A+BC$。

若 $F=\overline{AB+CD}$，则 $F'=\overline{(A+B)\cdot(C+D)}$。

【例 1-16】 试证明恒等式 $A+BC=(A+B)(A+C)$。

证明：根据对偶规则，$A+BC$ 的对偶式为

$$A(B+C)=AB+AC$$

$(A+B)(A+C)$的对偶式为

$$AB+AC$$

因对偶式相同，故 $A+BC$ 与$(A+B)(A+C)$相等，即

$$A+BC=(A+B)(A+C)$$

1.5 逻辑函数的几种常用描述方法及相互间的转换

如果以 n 个逻辑变量 A、B、C、… 作为输入，对其进行有限次逻辑运算的逻辑表达式 F，称为 n 变量的逻辑函数，写做

$$F=f(A,B,C,\cdots) \tag{1-25}$$

逻辑变量和逻辑函数的取值只可能是 0 和 1。

1.5.1　逻辑函数的几种常用描述方法

逻辑函数的描述方法很多，它可以用语言描述，亦可用逻辑表达式描述，还可用真值表、逻辑图、波形图、卡诺图等描述。

下面举一个简单的例子介绍前 4 种表示方法，卡诺图表示方法将在下一节介绍。

图 1-15 所示为一个举重裁判电路，比赛时主裁判掌握着开关 A，两名副裁判分别掌握着开关 B 和 C，当运动员举起杠铃时，裁判认为动作合格就合上开关，否则不合。比赛规则规定：只有当一名主裁判和 1 名以上副裁判认定运动员的动作合格时，试举才算成功。显然，指示灯 F 的状态是 A、B、C 3 个开关状态的函数。

1．真值表

为得到图 1-15 所示电路的输入变量的所有值和输出函数 *F* 的关系，设以 1 表示开关 A、B、C 的状态闭合，以 0 表示开关断开，以 1 表示指示灯亮，0 表示不亮，则可得真值表如表 1-18 所示。

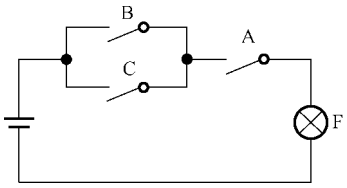


图 1-15　举重裁判电路

表 1-18　图 1-15 电路的真值表

输入			输出 *F*
A	*B*	*C*	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

2．逻辑表达式

逻辑表达式是用与、或、非等运算组合起来，表示逻辑函数与逻辑变量之间关系的逻辑代数式。

在图 1-15 所示的电路中，根据要求“B 和 C 中至少有一个合上”可以表示为(*B*+*C*)，而要求“同时还要求合上 A”，故应写做 *A*·(*B*+*C*)。因此得到输出的逻辑表达式为

$$F=A(B+C) \tag{1-26}$$

3．逻辑图

用与、或、非等逻辑符号表示逻辑函数中各变量之间的逻辑关系所得到的图形称为逻辑图。

将式（1-26）中所有的与、或、非运算符号用相应的逻辑符号代替，并按照逻辑运算的先后次序将这些逻辑符号连接起来，就得到图 1-15 所示电路对应的逻辑图，如图 1-16 所示。

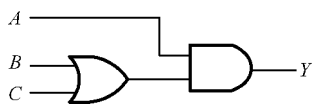


图 1-16　图 1-15 所示电路的逻辑图

4．波形图

如果将逻辑函数输入变量每一种可能出现的取值与对应的输出值按时间顺序依次排列起来，就得到了表示该逻辑函数的波形图。

图 1-17 的波形图是按照表 1-18 得到的：当 *ABC*=000 时，输出 *F*=0，……，直到 *ABC*=111 时，输出 *F*=1。也可以由式（1-26）计算得到。

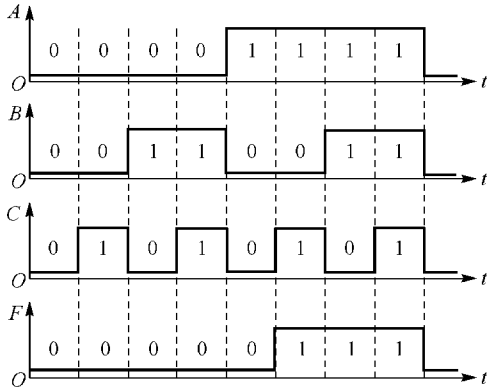


图 1-17　图 1-15 所示电路的波形图

1.5.2　不同描述方法之间的转换

由于逻辑函数具有多种描述方法，在逻辑问题的分析与设计中，经常需要将逻辑函数的某种描述方法转换为另一种描述方法，熟练掌握不同逻辑函数描述方法间的转换，有助于提高对逻辑问题的分析与设计能力。

经常用到的转换方式有：由真值表写出逻辑函数表达式、由逻辑函数表达式列出真值表、由逻辑函数表达式画出逻辑图和由逻辑图写出逻辑函数表达式等几种。

1．真值表与逻辑函数表达式的相互转换

（1）由真值表写出逻辑函数表达式

【例 1-17】 已知一个奇偶判别函数的真值表如表 1-19 所示，试写出它的逻辑函数表达式。

表 1-19　奇偶判别函数函数真值表

输入			输出 F
A	B	C	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

解：由真值表可见，在输入变量取值为以下 3 种情况时，F 等于 1：

A=0、B=1、C=1

A=1、B=0、C=1

A=1、B=1、C=0

而当 A=0、B=1、C=1 时，使乘积项 $\overline{A}BC$ =1；当 A=1、B=0、C=1 时，使乘积项 $A\overline{B}C$ =1；当 A=1、B=1、C=0 时，使乘积项 $AB\overline{C}$ =1。因此 F 的逻辑函数应当等于这 3 个乘积项之和，即

$$F = \overline{A}BC + A\overline{B}C + AB\overline{C}$$

通过例 1-17 可以总结出由真值表写出逻辑函数表达式的一般方法。

① 找出真值表中使逻辑函数 F=1 的那些输入变量取值的组合；

② 每组输入变量取值的组合对应一个乘积项，其中取值为 1 的写为原变量，取值为 0 的写为反变量；

③ 将这些乘积项相加，即得 F 的逻辑函数式。

（2）由逻辑函数表达式列出真值表

在由逻辑函数表达式列出函数的真值表时，只需将输入变量取值的所有组合状态逐一代入逻辑函数表达式，求出其对应的函数值，即可得到真值表。

【例 1-18】 已知逻辑函数 $F = A\overline{B} + B\overline{C} + C\overline{A}$，求它对应的真值表。

解：将 A、B、C 的各种取值逐一代入 F 式中计算，求出相应的函数值。当 ABC 取 000 时，F 为 0；当 ABC 取 001 时，F 为 1；……；当 ABC 取 111 时，F 为 0。得到表 1-20 所示的真值表。

2．逻辑函数表达式与逻辑图的相互转换

（1）由逻辑函数表达式画出逻辑图

在从已知的逻辑函数表达式转换为相应的逻辑图时，只要用逻辑图形符号代替逻辑函数式中的逻辑运算符号，并按运算优先顺序将它们连接起来，就可以得到所求的逻辑图了。

【例 1-19】 已知逻辑函数为 $F = A\overline{B}C + AB\overline{C} + ABC$，画出其对应的逻辑图。

解：由 $F = A\overline{B}C + AB\overline{C} + ABC$ 可以看出，三个乘积项 $A\overline{B}C$ 、$AB\overline{C}$ 、ABC 可由 3 个与门实现，它们的和运算可由或门实现，$\overline{B}$ 、$\overline{C}$ 可由非门实现。依据运算优先顺序把这些门电路连接起来，就可得到对应的逻辑电路图，如图 1-18 所示。

表 1-20　逻辑函数 $F = A\overline{B} + B\overline{C} + C\overline{A}$ 的真值表

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

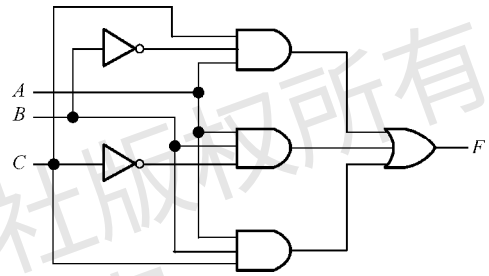


图 1-18　例 1-19 的逻辑电路图

（2）由逻辑图写出逻辑函数表达式

先从逻辑图的输入端到输出端逐级写出每个图形符号对应的输出逻辑表达式，再按照逻辑图前后级之间的逻辑关系进行组合运算，就可以得到电路中输出与输入之间的逻辑函数表达式了。

【例 1-20】 已知函数的逻辑图如图 1-19 所示，试求它的逻辑函数式。

解：从输入端 A、B、C 开始逐个写出每个图形符号输出端的逻辑式，得到

$$F = \overline{A \oplus B + \overline{B\overline{C}}}$$

将该式变换后得到

$$F = \overline{A \oplus B} \cdot B\overline{C}$$
$$= (AB + \overline{A} \cdot \overline{B})B\overline{C} = AB\overline{C}$$

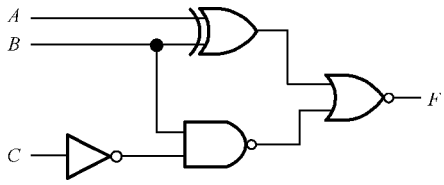


图 1-19　例 1-20 的逻辑图

3．波形图与真值表的相互转换

由已知的逻辑函数波形图求对应的真值表时，首先需要从波形图上找出每个时间段里输入变量与输出函数的取值，然后将这些输入、输出取值对应列表，就得到所求的真值表。

在将真值表转换为波形图时，只需将真值表中所有的输入变量与对应的输出变量取值依次排列画成以时间为横轴的波形，就得到了所求的波形图，如前面所做的那样。

【例 1-21】 已知逻辑函数 F 的波形如图 1-20 所示，试求该逻辑函数的真值表。

解：从 F 的波形图可以看出，输入变量 A、B、C 所有的取值组合均已出现了，因此，只要将 A、B、C 与 F 的取值对应列表，即可得表 1-21 的真值表。

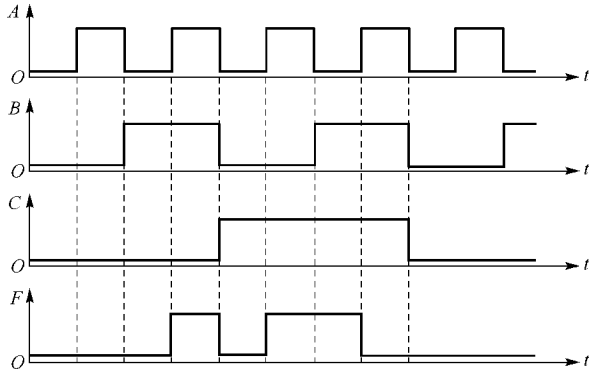


图 1-20　例 1-21 的波形图

表 1-21　例 1-21 的真值表

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

1.5.3　逻辑函数的建立及其描述

在生产和科学实验中，为了解决某个实际问题，必须研究其逻辑输出（因变量——“果”）和逻辑输入（自变量——“因”）相互之间的逻辑关系，从而得出相应的逻辑函数。一般来说，首先应根据提出的实际逻辑命题，确定哪些是输入逻辑变量（“因”），哪些是输出逻辑变量（“果”），然后研究它们之间的因果关系，列出其真值表，再根据真值表写出逻辑函数表达式。下面举一个实例来说明逻辑函数的建立步骤。

【例 1-22】 试设计一个电路，实现计算机上机管理控制电路。要求在管理员 C 同意时，学生 A 或 B 可以上机，但 A 具有优先上机权，A 不上机时，B 才可以上机。

解：输入变量有 3 个 A、B、C，设 A、B 为 1 时，表示要上机，为 0 时表示不要上机；当 C=1 时，表示管理员同意，C=0 时，表示管理员不同意。输出变量有两个 F_A 和 F_B，当 F_A 或 F_B 为 1 时表示能上机，为 0 时表示不能上机。列出真值表如表 1-22 所示。

F_A 和 F_B 的函数表达式为

$$F_A = A\overline{B}C + ABC = AC$$

$$F_B = \overline{A}BC$$

对应的逻辑电路图如图 1-21 所示。

表 1-22　例 1-22 的真值表

C	B	A	F_A	F_B
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

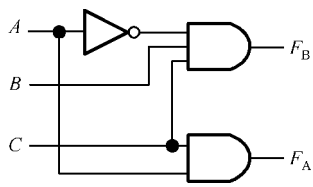


图 1-21　例 1-22 的逻辑图

1.6　逻辑函数的化简

同一逻辑函数可以采用不同的逻辑图来实现，而这些逻辑图所采用的器件的数量和种类可能会有所不同。因此，设计实际电路时，如果对逻辑函数进行化简，可以使设计出来的电路更简单，从而节省器件、降低成本、提高系统的可靠性，这对数字系统的工程设计来说具有重要的意义。

1.6.1　逻辑函数的最简形式和最简标准

一个逻辑函数可以有多种不同的逻辑表达式，例如，一个逻辑函数表达式为

$$F = AB + \overline{B}C$$

式中，AB 和 $\overline{B}C$ 两项都是由与运算把变量连接起来的，故称为与项（乘积项），然后由或运算将这两个与项连接起来，这种类型的表达式称为与-或逻辑表达式，或称为逻辑函数表达式的“积之和”形式。

一个与-或表达式可以转换为其他类型的函数表达式，例如，上面的与-或表达式经过变换，可以得到其与非-与非表达式、或-与表达式、或非-或非表达式以及与-或-非表达式等。

$$\begin{aligned}
F &= AB + \overline{B}C && \text{与-或表达式}\\
&= \overline{\overline{AB}\cdot\overline{\overline{B}C}} && \text{与非-与非表达式}\\
&= (A+\overline{B})(B+C) && \text{或-与表达式}\\
&= \overline{\overline{\overline{A}+\overline{\overline{B}}+\overline{B+C}}} && \text{或非-或非表达式}\\
&= \overline{\overline{AB}+\overline{B}\cdot\overline{\overline{C}}} && \text{与-或非表达式}\\
&= \overline{(\overline{A}+\overline{B})(B+\overline{C})} && \text{或-与-非表达式}
\end{aligned}$$

以上这些式子都是同一个函数的不同形式的最简表达式。最常用的有最简与或表达式和最简或与表达式。有了最简与或表达式以后，再用公式变换就可以得到其他类型的函数式。

与-或式的最简标准是：① 包含的与项个数最少；② 各与项中包含的变量个数最少。

或-与式的最简标准是：① 包含的或项个数最少；② 各或项中包含的变量个数最少。

常用的化简方法有公式法和卡诺图法两种。

1.6.2　逻辑函数的公式化简法

公式化简法就是反复运用逻辑代数的基本公式、常用公式和规则等，消去函数式中多余的乘积项和每个乘积项中多余的变量，以求得到逻辑函数表达式的最简形式。公式化简法没有固定步骤，现介绍几种常用的方法。

1．并项法

利用结合律 $AB + A\overline{B} = A$ 将两项合并成一项，并消去一个变量。例如

$$F_1 = ABC + AB\overline{C} + A\overline{B} = AB(C+\overline{C}) + A\overline{B} = AB + A\overline{B} = A(B+\overline{B}) = A$$

$$\begin{aligned}
F_2 &= A(BC + \overline{B}\overline{C}) + A(B\overline{C} + \overline{B}C) = ABC + A\overline{B}\overline{C} + AB\overline{C} + A\overline{B}C\\
&= AB(C+\overline{C}) + A\overline{B}(C+\overline{C}) = AB + A\overline{B} = A(B+\overline{B}) = A
\end{aligned}$$

2．吸收法

利用公式 $A+AB=A$ 消去多余的项。例如

$$F_1 = A\overline{C} + AB\overline{C}D(E+F) = A\overline{C}[1 + BD(E+F)] = A\overline{C}$$

$$F_2 = \overline{AB} + \overline{A}C + \overline{B}D = \overline{A} + \overline{B} + \overline{A}C + \overline{B}D = \overline{A} + \overline{B}$$

3．消去法

利用公式 $A + \overline{A}B = A + B$ 可将 $\overline{A}B$ 中的 $\overline{A}$ 消去。A、B 均可以是任何复杂的逻辑式。

$$F_1 = \overline{B} + ABC = \overline{B} + AC$$

$$F_2 = AB + \overline{A}C + \overline{B}C = AB + (\overline{A} + \overline{B})C = AB + \overline{AB}C = AB + C$$

4．消项法

利用公式 $AB + \overline{A}C + BC = AB + \overline{A}C$ 或 $AB + \overline{A}C + BCD = AB + \overline{A}C$ 可将 BC 或 BCD 项消去。其中，A、B、C、D 均可以是任何复杂的逻辑式。

$$\begin{aligned} F_1 &= ABC + \overline{A}D + \overline{C}D + BD = ABC + (\overline{A} + \overline{C})D + BD \\ &= AC \cdot B + \overline{AC}D + BD = ABC + \overline{AC}D \\ &= ABC + \overline{A}D + \overline{C}D \end{aligned}$$

$$F_2 = A\overline{B}C\overline{D} + \overline{A\overline{B}}E + \overline{A}C\overline{D}E = A\overline{B}C\overline{D} + \overline{A\overline{B}}E$$

5．配项法

利用公式 $A + \overline{A} = 1$、$A + A = A$、$A \cdot A = A$、$1 + A = 1$ 等基本公式给某些函数配上适当的项，进而可消去原函数中的某些项或变量。例如

$$\begin{aligned} F_1 &= \overline{A}B + A\overline{B} + AB = \overline{A}B + AB + A\overline{B} + AB \\ &= (\overline{A} + A)B + A(\overline{B} + B) = A + B \\ F_2 &= AB + \overline{A\overline{C}} + B\overline{C} = AB + \overline{A\overline{C}} + (A + \overline{A})B\overline{C} \\ &= AB + \overline{A\overline{C}} + AB\overline{C} + \overline{A}B\overline{C} = AB + AB\overline{C} + \overline{A\overline{C}} + \overline{A}B\overline{C} \\ &= AB + \overline{A\overline{C}} \end{aligned}$$

公式化简法不仅需要熟悉逻辑代数的基本公式和常用公式，而且还要灵活运用这些公式，需要较高的技巧性。否则，化简过程中就容易走弯路，甚至不能化到最简结果（多数情况是很难判断结果是否最简以及是否正确）。因此，用公式化简法时要注意以下几点。

（1）若原函数不是与或表达式，应先将其转换成与或表达式的形式，然后再进行化简。

（2）尽可能先使用并项法、吸收法、消去法、消项法等简单方法进行化简，若这些方法无效，再考虑使用配项法。

（3）化简后的表达式不是唯一的，但要求是最简的（但与卡诺图法相比很难判断结果是否最简）。

（4）根据化简的需要，适当添加相应的多余项，可有利于化简过程。

下面通过例子进一步说明综合应用公式法的化简过程。

【例 1-23】 化简函数 $F = ABC\overline{D} + ABD + BC\overline{D} + ABC + BD + B\overline{C}$。

解：

$$\begin{aligned} F &= ABC\overline{D} + ABD + BC\overline{D} + ABC + BD + B\overline{C} \\ &= ABC + ABC\overline{D} + BD + ABD + BC\overline{D} + B\overline{C} \\ &= ABC + BD + BC\overline{D} + B\overline{C} && \text{（吸收法）} \\ &= B(AC + \overline{C}) + B(D + C\overline{D}) \\ &= B(A + \overline{C}) + B(D + C) && \text{（消去法）} \\ &= AB + B\overline{C} + BC + BD \\ &= AB + B + BD && \text{（并项法）} \\ &= B && \text{（吸收法）} \end{aligned}$$

对初学者来说，很难保证化简结果达到最简。只有当逻辑函数非常简单时（一般是少于 3 个变量），用公式法化简才比较简便。

总之，公式化简法不直观，一般不容易判断结果是否最简。因此，人们普遍采用卡诺图化简法来化简逻辑函数，它是一种具有统一规则的逻辑函数化简方法。在 1.6.4 节中将讨论此种方法。

1.6.3　逻辑函数的两种标准形式

在讲述逻辑函数的标准形式之前，先介绍一下最小项和最大项的概念，然后再介绍逻辑函数的最小项之和及最大项之积这两种标准形式。

1．最小项和最大项

（1）最小项

在 n 个变量组成的乘积项中，若每个变量都以原变量或以反变量的形式出现且仅出现一次，那么该乘积项称做 n 变量的一个最小项。

例如，A、B、C 3 个变量的最小项有：$\overline{A}\overline{B}\overline{C}$ 、$\overline{A}\overline{B}C$ 、$\overline{A}B\overline{C}$ 、$\overline{A}BC$ 、$A\overline{B}\overline{C}$ 、$A\overline{B}C$ 、$AB\overline{C}$ 、ABC 。它们都含有 3 个变量，而每个变量都以原变量或反变量形式在一个乘积项中出现一次，故三变量的最小项共有 2^3=8 个。同理，二变量的最小项共有 2^2=4 个；四变量的最小项有 2^4=16 个；n 变量的最小项共有 2^n 个。

输入变量的每一组取值都使一个对应的最小项的值等于 1。例如，在三变量 A、B、C 的最小项中，当 A=0、B=1、C=1 时，$\overline{A}BC=1$。如果把 $\overline{A}BC$ 的取值 011 看做一个二进制数，那么它所表示的十进制数就是 3。为了今后使用的方便，将 $\overline{A}BC$ 这个最小项记做 m_3。按照这一约定，就得到了三变量最小项的编号表，如表 1-23 所示。

表 1-23　三变量最小项的编号表

最小项	变量取值			编号
	A	B	C	
$\overline{A}\overline{B}\overline{C}$	0	0	0	m_0
$\overline{A}\overline{B}C$	0	0	1	m_1
$\overline{A}B\overline{C}$	0	1	0	m_2
$\overline{A}BC$	0	1	1	m_3
$A\overline{B}\overline{C}$	1	0	0	m_4
$A\overline{B}C$	1	0	1	m_5
$AB\overline{C}$	1	1	0	m_6
ABC	1	1	1	m_7

由表 1-23 可知，三变量 A、B、C 的最小项记做 $m_0 \sim m_7$，同理四变量 A、B、C、D 的最小项记做 $m_0 \sim m_{15}$。

从最小项的定义出发可以证明它具有如下性质：

① 在任何一组输入变量的取值下，只有一个最小项的值为 1，其余最小项的值均为 0；

② 任何两个不同的最小项的乘积为 0；

③ 任何一组变量取值下，全部最小项之和为 1。

（2）最大项

在 n 个变量组成的或项中，若每个变量都以原变量或以反变量的形式出现且仅出现一次，那么该或项称做 n 变量的一个最大项。

例如，A、B、C 3 个变量的最大项有：$(\overline{A}+\overline{B}+\overline{C})$ 、$(\overline{A}+\overline{B}+C)$ 、$(\overline{A}+B+\overline{C})$ 、$(\overline{A}+B+C)$ 、$(A+\overline{B}+\overline{C})$ 、$(A+\overline{B}+C)$ 、$(A+B+\overline{C})$ 、$(A+B+C)$ 。它们都含有 3 个变量，而每个变量都以原变量或反变量的形式在一个或项中出现一次，故三变量的最大项共有 2^3=8 个。同理，二变量的最大项共有 2^2=4 个；四变量的最大项有 2^4=16 个；n 变量的最大项共有 2^n 个。

输入变量的每一组取值都使一个对应的最大项的值等于 0。例如，在三变量 A、B、C 的最大项中，当 A=0、B=1、C=1 时，$(A+\overline{B}+\overline{C})=0$。如果将最大项为 0 的 ABC 取值视为一个二进制数，并以其对应的十进制数给最大项编号，则 $(A+\overline{B}+\overline{C})$ 可记做 M_3。由此得到三变量最大项的编号表，如表 1-24 所示。

从最大项的定义出发同样可以得到它的主要性质：

① 在任何一组输入变量的取值下，只有一个最大项的值为 0，其余最大项的值均为 1；

表 1-24 三变量最大项的编号表

最大项	使最大项为 0 的变量取值			编号
	A	B	C	
$A+B+C$	0	0	0	M_0
$A+B+\overline{C}$	0	0	1	M_1
$A+\overline{B}+C$	0	1	0	M_2
$A+\overline{B}+\overline{C}$	0	1	1	M_3
$\overline{A}+B+C$	1	0	0	M_4
$\overline{A}+B+\overline{C}$	1	0	1	M_5
$\overline{A}+\overline{B}+C$	1	1	0	M_6
$\overline{A}+\overline{B}+\overline{C}$	1	1	1	M_7

② 任何两个不同的最大项的和为 1；

③ 任何一组变量取值下，全部最大项之积为 0。

将表 1-23 与表 1-24 加以对比可发现，最大项和最小项之间关系为

$$M_i = \overline{m_i} \qquad (1\text{-}27)$$

例如，$m_3 = \overline{A}BC$，则

$$\overline{m_3} = \overline{\overline{A}BC} = A + \overline{B} + \overline{C} = M_3$$

2. 逻辑函数的标准与或表达式

每个乘积项都是最小项的与或表达式，称为标准与或表达式，也称为最小项之和表达式。

一个逻辑函数表示成标准与或表达式有两种方法。

（1）从真值表求标准与或表达式

① 找出使逻辑函数 F 为 1 的变量取值组合；

② 写出使函数 F 为 1 的变量取值组合对应的最小项；

③ 将这些最小项相或，即得到标准与或表达式。

例 1-17 中由真值表写出的逻辑函数表达式即是标准与或表达式。

（2）从一般逻辑表达式求标准与或表达式

首先将给定的逻辑函数式化为若干乘积项之和的形式，然后利用公式 $A+\overline{A}=1$ 将每个乘积项中缺少的因子补全，这样就可以将与或的形式化为最小项之和的形式，即标准与或表达式。

【例 1-24】 写出逻辑函数 $F = A\overline{B}\overline{C}D + BCD + \overline{A}D$ 的标准与或表达式。

解：

$$\begin{aligned} F &= A\overline{B}\overline{C}D + BCD + \overline{A}D \\ &= A\overline{B}\overline{C}D + (A+\overline{A})BCD + \overline{A}(B+\overline{B})(C+\overline{C})D \\ &= A\overline{B}\overline{C}D + ABCD + \overline{A}BCD + \overline{A}B(C+\overline{C})D + \overline{A}\overline{B}(C+\overline{C})D \\ &= A\overline{B}\overline{C}D + ABCD + \overline{A}BCD + \overline{A}B\overline{C}D + \overline{A}\overline{B}CD + \overline{A}\overline{B}\overline{C}D \\ &= A\overline{B}\overline{C}D + ABCD + \overline{A}BCD + \overline{A}B\overline{C}D + \overline{A}\overline{B}CD + \overline{A}\overline{B}\overline{C}D \end{aligned}$$

或写做

$$F(A,B,C,D) = m_1 + m_3 + m_5 + m_7 + m_9 + m_{15} = \sum m(1,3,5,7,9,15)$$

3. 逻辑函数的标准或与表达式

每个或项都是最大项的或与表达式，称为标准或与表达式，也称为最大项之积表达式。

从逻辑函数真值表求标准或与表达式的方法为：

（1）找出使逻辑函数 F 为 0 的行；

（2）对于 F=0 的行，写出对应的最大项；

（3）将这些最大项相与，即得到标准或与表达式。

表 1-18 的真值表重新列出在表 1-25 中。

写出其标准或与表达式为

$$F = (A+B+C)(A+B+\overline{C})(A+\overline{B}+C)(A+\overline{B}+\overline{C})(\overline{A}+B+C)$$

或写做

表 1-25 三变量真值表

输入			输出 F
A	B	C	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$F = M_0 \cdot M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot M_5 = \prod M(0,1,2,3,4,5)$$

【例 1-25】 将逻辑函数 $F = A\overline{B} + AC$ 化为标准或与表达式。

解：先将函数化为标准与或表达式

$$\begin{aligned} F &= \overline{A}B + AC \\ &= \overline{A}B(C + \overline{C}) + A(B + \overline{B})C \\ &= \overline{A}BC + \overline{A}B\overline{C} + ABC + A\overline{B}C \\ &= m_2 + m_3 + m_5 + m_7 = \sum m(2,3,5,7) \end{aligned}$$

再转换成标准或与表达式为

$$F = \prod M(0,1,4,6)$$

因逻辑函数的真值表是唯一的，所以逻辑函数的标准与或表达式和或与表达式都是唯一的。

1.6.4　逻辑函数的卡诺图化简法

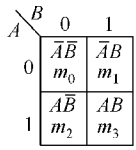
1．逻辑函数的卡诺图表示法

前面已经知道，n 个逻辑变量可以组成 2^n 个最小项。在这些最小项中，如果两个最小项仅有一个因子不同，而其余因子均相同，则称这两个逻辑最小项为逻辑相邻项。如三变量的最小项 ABC 、$AB\overline{C}$ 仅有一个因子 C 不同，其余因子相同，它们是逻辑相邻项。

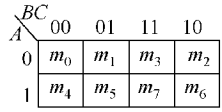
为了表示出最小项之间这种逻辑相邻关系，美国工程师卡诺（Karnaugh）设计了一种最小项方格图，他把逻辑相邻项安排在位置相邻的方格中。按此规则排列起来的最小项方格图称为卡诺图。

例如，两个变量 A、B 有 4 个最小项：$\overline{A}\overline{B}$ 、$\overline{A}B$ 、$A\overline{B}$ 、AB ，分别记做 m_0 、m_1 、m_2 、m_3 。它们的卡诺图如图 1-22(a)所示，显然图中上下、左右之间的最小项都是逻辑相邻项。A、B 变量标注在卡诺图的左上角，卡诺图的左侧和上方标注的 0 和 1 表示使对应方格内的最小项为 1 的变量取值。同时，这些 0 和 1 组成的二进制数所对应的十进制数大小也就是对应的最小项的编号。

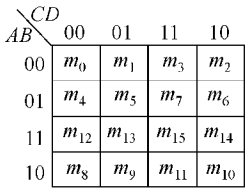
图 1-22(b)、图 1-22(c)、图 1-22(d)分别是三变量、四变量、五变量最小项的卡诺图。图中不仅相邻方格的最小项是逻辑相邻项，而且上下、左右相对的方格也是逻辑相邻项。为了保证这种相邻性，卡诺图左边和上边的数码不能按自然二进制数从小到大的顺序排列，而必须按循环格雷码排列。



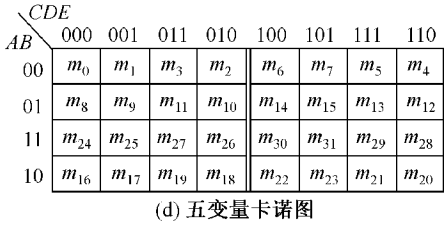
(a) 二变量卡诺图



(b) 三变量卡诺图



(c) 四变量卡诺图



(d) 五变量卡诺图

图 1-22　二变量到五变量的最小项卡诺图

在变量数大于等于 5 后，仅用几何图形在两维空间的相邻性来表示逻辑相邻性已经不够了。例如，在图 1-22(d)所示的五变量最小项的卡诺图中，除了几何位置相邻的最小项具有逻辑相邻性以外，以图中双竖线为轴左右对称位置上的两个最小项也具有逻辑相邻性。当包含的变量数多于 5 个时，卡诺图则不易画出，相邻性也失去了直观的特点。

任何一个逻辑函数都可以用最小项之和的形式表示，所以也就可以用卡诺图来表示。具体方法是：首先根据逻辑函数所包含的变量数目，画出相应的卡诺图；然后将函数式中包含的最小项，在卡诺图对应方格中填 1；在其余位置上填入 0，就得到了表示该函数的卡诺图。

【例 1-26】 画出逻辑函数 $F(A,B,C,D)=\sum m(0,1,2,3,4,8,10,11,14,15)$ 的卡诺图。

解：对逻辑函数表达式中的各最小项，在卡诺图相应小方格内填入 1，其余填入 0，即可得图 1-23 所示的卡诺图。

【例 1-27】 画出逻辑函数

$$F(A,B,C,D)=(\overline{A}+\overline{B}+\overline{C}+\overline{D})(\overline{A}+\overline{B}+C+\overline{D})(\overline{A}+B+\overline{C}+D)$$
$$(A+\overline{B}+\overline{C}+D)(A+B+C+D)$$

的卡诺图。

解：该逻辑函数已是最大项之积的表示形式

$$F(A,B,C,D)=M_{15}\cdot M_{13}\cdot M_{10}\cdot M_6\cdot M_0$$

对表达式中的各最大项，在卡诺图相应小方格内填入 0，其余填入 1，即可得图 1-24 所示的卡诺图。

【例 1-28】 用卡诺图表示以下逻辑函数。

$$F=A\overline{B}+\overline{A}B\overline{D}+ACD+\overline{A}\overline{B}\overline{C}D$$

解：先将逻辑函数化为最小项之和的形式

$$\begin{aligned}F&=A\overline{B}+\overline{A}B\overline{D}+ACD+\overline{A}\overline{B}\overline{C}D\\&=A\overline{B}(C+\overline{C})(D+\overline{D})+\overline{A}B(C+\overline{C})\overline{D}+A(B+\overline{B})CD+\overline{A}\overline{B}\overline{C}D\\&=A\overline{B}(CD+\overline{C}D+C\overline{D}+\overline{C}\overline{D})+\overline{A}BC\overline{D}+\overline{A}B\overline{C}\overline{D}+ABCD+A\overline{B}CD+\overline{A}\overline{B}\overline{C}D\\&=A\overline{B}CD+A\overline{B}\overline{C}D+A\overline{B}C\overline{D}+A\overline{B}\overline{C}\overline{D}+\overline{A}BC\overline{D}+\overline{A}B\overline{C}\overline{D}+ABCD+\overline{A}\overline{B}\overline{C}D\\&=\sum m(1,4,6,8,9,10,11,15)\end{aligned}$$

然后再参照前面的方法画出卡诺图，如图 1-25 所示。

AB \ CD	00	01	11	10
00	1	1	1	1
01	1	0	0	0
11	0	0	1	1
10	1	0	1	1

图 1-23　例 1-26 的卡诺图

AB \ CD	00	01	11	10
00	0	1	1	1
01	1	1	1	0
11	1	0	0	1
10	1	1	1	0

图 1-24　例 1-27 的卡诺图

AB \ CD	00	01	11	10
00	0	1	0	0
01	1	0	0	1
11	0	0	1	0
10	1	1	1	1

图 1-25　例 1-28 的卡诺图

2．用卡诺图化简逻辑函数

前面我们介绍过，卡诺图的相邻项仅有一个因子不同，其余因子相同，所以可以根据公式

$AB+A\overline{B}=A$，将卡诺图的两个逻辑相邻项之和合并成一项，并消去一个因子。因此，利用卡诺图可以形象、直观地找到逻辑相邻项并将其合并，得到简化的函数式。这种利用卡诺图化简函数的方法称为卡诺图化简法或图形化简法。

（1）合并最小项的规则

在逻辑函数的卡诺图中，两个相邻的最小项可以合并成一项，并消去那个不相同的因子，合并的结果中只剩下公共因子。

图 1-26(a)中以四变量的卡诺图为例画出了两个最小项相邻的几种可能情况，如 m_5 和 m_{13} 是相邻项，故可合并为

$$m_5+m_{13}=\overline{A}B\overline{C}D+AB\overline{C}D=(\overline{A}+A)B\overline{C}D=B\overline{C}D$$

合并后剩下了公共因子 $B\overline{C}D$。

4 个相邻的最小项并排成矩形组，可合并为一项，并消去两个因子，合并后的结果中只剩下公共因子。

图 1-26(b)中以四变量的卡诺图为例画出了 4 个最小项相邻的几种可能情况，如 m_5、m_7、m_{13} 和 m_{15} 是相邻项，合并后得到

$$\begin{aligned}m_5+m_7+m_{13}+m_{15}&=\overline{A}B\overline{C}D+\overline{A}BCD+AB\overline{C}D+ABCD\\&=\overline{A}BD(\overline{C}+C)+ABD(\overline{C}+C)\\&=(\overline{A}+A)BD=BD\end{aligned}$$

合并后剩下了 4 个最小项的公共因子 BD。

8 个相邻的最小项并排成矩形组，可合并为一项，并消去 3 个因子，合并后的结果中只剩下公共因子。

图 1-26(c)所示为 8 个最小项相邻的几种可能情况。如上面两行的 8 个最小项是相邻项，合并后只剩下一项 $\overline{A}$，其他的因子都被消去了。

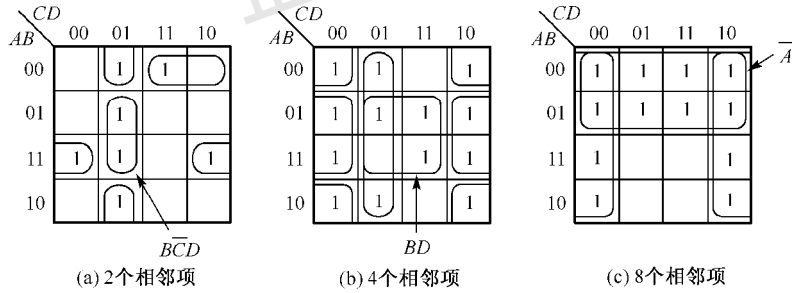


图 1-26　四变量卡诺图的几种相邻项

由此得到合并最小项的一般规则：

① 能够合并的最小项数必须是 2 的整数次幂，即 1、2、4、8、…；

② 要合并的相应方格必须排列成矩形或正方形。

（2）用卡诺图化简函数的步骤

用卡诺图化简函数的步骤如下：

① 画出逻辑函数的卡诺图；

② 按照上述合并最小项的规则，将可以合并的最小项圈起来，没有相邻项的最小项单独画圈；

③ 将所有圈对应的乘积项相加。

上述②中画圈的原则是：

① 包围圈内的方格数要尽可能多，包围圈的数目要尽可能少；

② 同一方格可以被不同的包围圈重复包围，但新增包围圈中一定要有新的方格，否则该包围圈为多余。

【例 1-29】 用卡诺图化简法将下式化简为与或函数式。

$$F = A\overline{C} + \overline{A}C + B\overline{C} + \overline{B}C$$

解： 首先画出 F 的卡诺图，如图 1-27 所示。事实上，在画卡诺图时，可以不通过将 F 化为最小项之和的形式。例如，式中的 $A\overline{C}$ 一项包含了所有含有 $A\overline{C}$ 因子的最小项，而不管另一个因子是 B 还是 $\overline{B}$ 。从另一个角度讲，也可以理解为 $A\overline{C}$ 是 $AB\overline{C}$ 和 $A\overline{B}\overline{C}$ 两个最小项相加合并的结果。因此，在填写 F 的卡诺图时，可以直接在卡诺图上所有对应 A=1、C=0 的空格里填入 1。按照这种方法，就可以省去将 F 化为最小项之和这一步骤了。

然后把可能合并的最小项圈出。由图 1-27 可见，有两种可取的合并最小项的方案。按图 1-27(a)的圈法，可得

$$F = A\overline{B} + B\overline{C} + \overline{A}C$$

按图 1-27(b)的圈法，可得

$$F = A\overline{C} + \overline{A}B + \overline{B}C$$

此例说明，一个逻辑函数的化简结果可能不是唯一的，但二者都是最简的。

【例 1-30】 用卡诺图化简法将下式化为最简与或表达式。

$$F = \overline{A}\overline{B}\overline{C} + \overline{A}B\overline{D} + \overline{A}C\overline{D} + AB\overline{C}D + \overline{B}\overline{D}$$

解： 首先画出 F 的卡诺图，如图 1-28 所示。然后画圈，m_0、m_1 圈在一起，合并后为 $\overline{A}\overline{B}\overline{C}$；$m_2$、$m_6$ 圈在一起，合并后为 $\overline{A}C\overline{D}$；4 个角上的方格圈在一起，合并后为 $\overline{B}\overline{D}$；独立的圈为 $AB\overline{C}D$ 。故得到化简后的表达式为

$$F = AB\overline{C}D + \overline{A}\overline{B}\overline{C} + \overline{A}C\overline{D} + \overline{B}\overline{D}$$

上面的例子都是通过合并卡诺图中的 1（即圈 1 法）来求得化简结果的，得到的是与或表达式。如果卡诺图中的 1 较多，0 较少时，也可以通过合并卡诺图中的 0（即圈 0）先求出 $\overline{F}$ 的化简结果，然后再将 $\overline{F}$ 求反而得到 F。

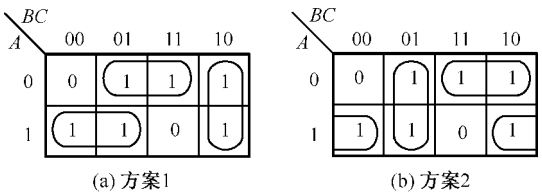


图 1-27　例 1-29 的卡诺图

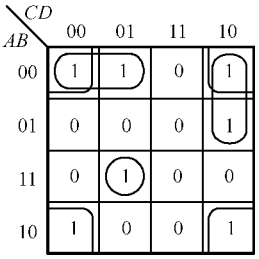


图 1-28　例 1-30 的卡诺图

【例 1-31】 用卡诺图化简法化简如下逻辑函数。

$$F = ABC + ABD + A\overline{C}D + \overline{C}\overline{D} + A\overline{B}C + \overline{A}C\overline{D}$$

解： 首先画出 F 的卡诺图，如图 1-29 所示。

然后画圈，本例的卡诺图中由于 0 较少，所以采用圈 0 的方法把 4 个 0 全部圈起来，得到

$$\overline{F} = \overline{A}D$$

则

$$F = \overline{\overline{F}} = \overline{\overline{A}D} = A + \overline{D}$$

另外，如果需要将函数化简为与或非表达式时，也采用圈 0 的方法。

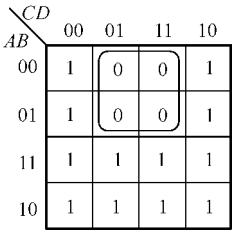


图 1-29　例 1-31 的卡诺图

1.6.5　具有无关项的逻辑函数的化简法

1．约束项、任意项和无关项

在分析某些具体的逻辑函数时，经常会遇到这样的情况，即输入变量的取值不是任意的。对输入变量的取值所加的限制称为约束，同时把这一组变量称为具有约束的一组变量。

例如，有 3 个逻辑变量 A、B、C，它们分别表示一台电动机的正转、反转和停止的命令，A=1 表示正转，B=1 表示反转，C=1 表示停止。因为电动机在任何时刻只能执行其中的一个命令，所以不允许两个以上的变量同时为 1，故 ABC 的取值只可能是 001、010、100 之中的某一种，而不能是 000、011、101、110、111 中的任何一种。因此，A、B、C 是一组具有约束的变量。

通常用约束条件来描述约束的具体内容。显然，用上面的这样一段文字叙述约束条件是很不方便的，最好能用简单、明了的逻辑语言表述约束条件。

由于每一组输入变量的取值都使一个、而且仅有一个最小项的值为 1，所以当限制某些输入变量的取值不能出现时，可以用它们对应的最小项恒等于 0 来表示。这样，上面例子中的约束条件可以表示为

$$\begin{cases} \overline{A}\overline{B}\overline{C} = 0 \\ \overline{A}BC = 0 \\ A\overline{B}C = 0 \\ AB\overline{C} = 0 \\ ABC = 0 \end{cases}$$

或写成

$$\overline{A}\overline{B}\overline{C} + \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC = 0$$

同时，把这些恒等于 0 的最小项叫做约束项。

有时还会遇到另外一种情况，就是在输入变量的某些取值下函数值是 1 或 0 皆可，并不影响电路的功能。在这些变量取值下，其值等于 1 的那些最小项称为任意项。例如，在设计一个逻辑判断电路时，要求判断 1 位十进制数是奇数还是偶数，当十进制数为奇数时，函数值为 1，反之为 0。用 4 位二进制码组成 8421BCD 码时，4 位二进制码共有 16 种变量组合，而 8421BCD 码只选中其中的 0000～1001 等 10 种变量组合来代表 0～9 共 10 个十进制数，其余 6 种变量组合 1010～1111 对应的函数值可以是任意的，为 0 为 1 都可以。这 6 种变量组合构成的最小项就是任意项。

在存在约束项的情况下，由于约束项的值始终等于 0，所以既可以把约束项写进逻辑函数式中，也可以把约束项从逻辑函数式中删掉，而不影响函数值。同样，既可以把任意项写入函数式中，也可以不写进去，因为输入变量的取值使这些任意项为 1 时，函数值是 1 或 0 皆可。

因此，又把约束项和任意项统称为逻辑函数式中的无关项。这里所说的无关是指是否把这些最小项写入逻辑函数式无关紧要，可以写入，也可以删除。

之前曾经讲到，在用卡诺图表示逻辑函数时，首先将函数化为最小项之和的形式，然后在卡诺图中将这些最小项对应的位置上填入 1，其他位置上填入 0。既可以认为无关项包含于函数式中，也可以认为不包含在函数式中，那么在卡诺图中对应的位置上既可以填入 1，也可以填入 0。为此通常在卡诺图中用符号“×”、“－”或“Φ”来表示无关项。在化简逻辑函数时，无关项“×”可视为 1，也可视为 0。

2．无关项在逻辑函数化简中的应用

化简具有无关项的逻辑函数时，如果能合理利用这些无关项，一般都可得到更加简单的化简结果。

为达到此目的，加入的无关项应与函数式中尽可能多的最小项（包括原有的最小项和已写入的无关项）具有逻辑相邻性。

合并最小项时，究竟把卡诺图上的“×”作为 1（即认为函数式中包含了这个最小项）还是作为 0（即认为函数式中不包含这个最小项）对待，应以得到的相邻最小项矩形组合最大、而且矩形组合数目最少为原则。

【例 1-32】 用卡诺图法求下列函数的最简与或式。

$$F=\overline{A}BCD+\overline{A}B\overline{C}D+\overline{A}BC\overline{D}+A\overline{B}CD+AB\overline{C}D+ABC\overline{D}$$

给定约束条件为

$$\overline{A}\overline{B}\overline{C}D+\overline{A}\overline{B}C\overline{D}+\overline{A}B\overline{C}\overline{D}+A\overline{B}\overline{C}\overline{D}+AB\overline{C}\overline{D}=0$$

解：首先画出函数的卡诺图如图 1-30 所示。

将所有的约束项都看做 1，画圈如图 1-30 所示，则最简与或式为

$$F=A+B\overline{C}\overline{D}+\overline{B}C\overline{D}+\overline{B}\overline{C}D$$

【例 1-33】 某逻辑电路输入信号 A、B、C、D 为 8421BCD 码，当码值为 1、3、5、7、9 时，输出函数 F 为 1。求该电路输出函数的最简与或表达式。

解：首先列出电路的真值表如表 1-26 所示。

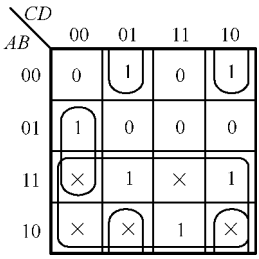


图 1-30　例 1-32 的卡诺图

表 1-26　例 1-33 的真值表

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	×
1	0	1	1	×
1	1	0	0	×
1	1	0	1	×
1	1	1	0	×
1	1	1	1	×

当输入为 8421BCD 码的 6 个输入组合 1001、1010、…、1111 时，输出函数为任意项，既可为 1，也可为 0。其卡诺图如图 1-31 所示。

将无关项 m_{11}、m_{13}、m_{15} 看做 1，m_{10}、m_{12}、m_{14} 看做 0，画圈如图 1-31 所示，则最简与或式为

$$F=D$$

本例若不利用无关项，化简结果为

$$F=\overline{A}D+\overline{B}\overline{C}D$$

可见，利用无关项可以使化简结果进一步简化。

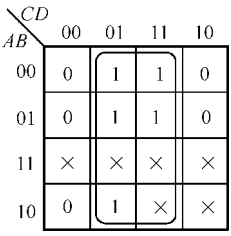


图 1-31　例 1-33 的卡诺图

1.6.6　多输出变量的卡诺图化简法

前述的电路只有一个输出端，而实际电路常常有两个或两个以上的输出端。化简多输出函数时，不能单纯地去追求各单一函数的最简式，因为这样做并不一定能保证整个系统最简，应该统一考虑，尽可能多地利用公共项。

【例 1-34】 试用卡诺图化简如下多输出函数。

$$\begin{cases}F_1(A,B,C)=\sum m(1,3,4,5,7)\\ F_2(A,B,C)=\sum m(3,4,7)\end{cases}$$

解：如果先按每个函数各自的卡诺图画圈，如图 1-32 所示，化简得

$$F_1=A\overline{B}+C$$

$$F_2=A\overline{B}\overline{C}+BC$$

可用图 1-33 所示的电路图来实现。

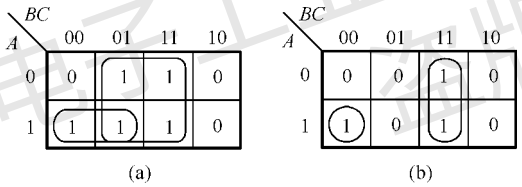


图 1-32　各函数用卡诺图独立化简

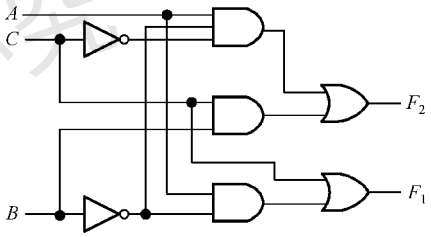


图 1-33　各函数独立化简的电路图

再将两个函数视为一个整体，对卡诺图画圈，如图 1-34 所示。化简得

$$F_1=A\overline{B}\overline{C}+C$$

$$F_2=A\overline{B}\overline{C}+BC$$

可用图 1-35 所示的电路图来实现。

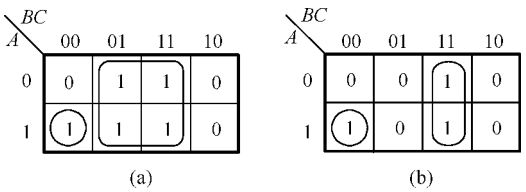


图 1-34　各函数用卡诺图整体考虑化简

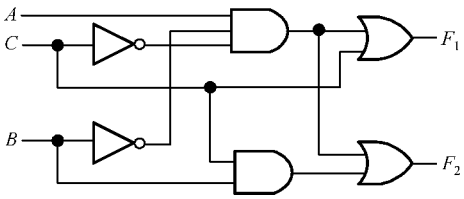


图 1-35　各函数整体考虑化简的电路图

可以看到，对各函数整体考虑化简后的电路由于有公共的电路部分，整个电路更简单。

习　题

1.1　将下列二进制整数转换为等值的十进制数。

（1）$(01010)_B$；　（2）$(10101)_B$；　（3）$(01010101)_B$；　（4）$(10101010)_B$。

1.2　将下列二进制整数转换为等值的八进制数和十六进制数。

（1）$(001.010)_B$；　（2）$(1101.1010)_B$；　（3）$(1101.0111)_B$；　（4）$(101101.010)_B$

1.3　将下列十六进制数转换为等值的二进制数。

（1）$(7F)_H$；　（2）$(01.56)_H$；　（3）$(8B.EF)_H$；　（4）$(A2.4D)_H$。

1.4　将下列十进制整数转换为等值的二进制数和十六进制数。

（1）$(16)_D$；　（2）$(126)_D$；　（3）$(78)_D$；　（4）$(253)_D$。

1.5　写出下列二进制数的原码、反码和补码。

（1）$(+1010)_B$；　（2）$(+10101)_B$；　（3）$(-0101)_B$；　（4）$(-01010)_B$。

1.6　用二进制补码加、减法运算规则计算下列各式。式中的 4 位二进制数是不带符号位的数值位（提示：所用补码的有效位应能足够表示其结果的最大数值位）。

（1）42+21；　（2）42–21；

（3）21+42；　（4）21–42；

（5）0111–1001；　（6）1010–1100；

（7）–0111–1001；　（8）–1010–1100。

1.7　证明下列逻辑式成立（方法不限）。

（1）$(A+\overline{B})(\overline{A}+\overline{B}+\overline{C})=A\overline{C}+\overline{B}$；

（2）$\overline{A(B+C)+\overline{A}\overline{B}}=\overline{B}(\overline{A}+\overline{C})$；

（3）$(\overline{A}+\overline{B}+\overline{C})(B+C)(A+\overline{B})=AB\overline{C}+\overline{B}C$；

（4）$\overline{AC+\overline{A}BC+\overline{B}\overline{C}}+AB\overline{C}=\overline{C}$。

1.8　用反演规则或对偶规则写出下列函数的反函数和对偶函数。

（1）$Y(A,B,C)=(A+B)(B+\overline{C})+AC$；

（2）$Y(A,B,C,D)=\overline{A\overline{B}C+BC\overline{D}}(A+C)+BD$；

（3）$Y(A,B,C,D)=[\overline{(A+D)\overline{AC}}+\overline{A}B\overline{D}](\overline{A+C}+BD)$；

（4）$Y(A,B,C,D)=\overline{(A\oplus C)(B+\overline{D})}(BD+AC)$。

1.9　已知逻辑函数的真值表如表 P1-1(a)、表 P1-1(b)所示，试写出对应的逻辑函数式。

表 P1-1(a)

A	*B*	*C*	*Y*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

表 P1-1(b)

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

1.10　列出下列逻辑函数的真值表。

（1）$F=(A+\overline{B})(\overline{A}+\overline{B}+\overline{C})$；

（2）$Y=BC\overline{D}+C+\overline{A}D$。

1.11　写出图 P1-1(a)、图 P1-1(b)所示电路的输出逻辑函数式。

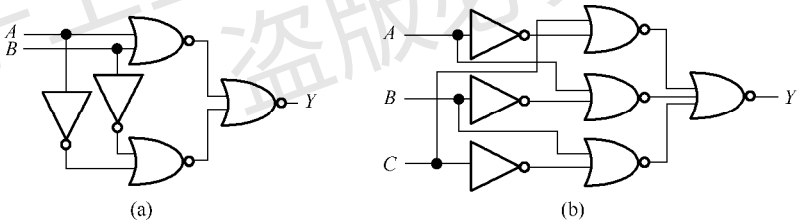


图 P1-1

1.12　已知逻辑函数 F 的波形图如图 P1-2 所示，试写出该逻辑函数的真值表和逻辑函数式。

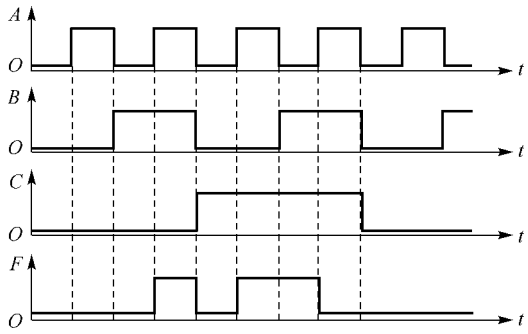


图 P1-2

1.13　将下列各函数式化为最小项之和的形式。

（1）$F(A,B,C)=A\overline{C}+\overline{B}C+\overline{\overline{A}+B+C}(A+\overline{C})$；

（2）$F(X,Y,Z)=(\overline{X}+YZ)(\overline{Y}+X\overline{Z})+XYZ$；

（3）$F(A,B,C,D)=A\overline{B}C+B\overline{C}D+C\overline{D}A+\overline{A}B\overline{C}$；

（4）$F(W,X,Y,Z)=W\overline{X}(Y+\overline{Z})+(\overline{W}+X)\overline{Y}Z+\overline{\overline{X}Y}$。

1.14　将下列各函数式化为最大项之积的形式。

（1）$F(A,B,C)=A\overline{B}C+\overline{A}C(B+\overline{C}A)$；

（2）$F(X,Y,Z)=\overline{X}\overline{Y}\overline{Z}+\overline{(\overline{X}+Y)}(\overline{Y}+\overline{Z})$；

（3）$F(A,B,C,D)=A\overline{C}+\overline{B}D+\overline{AB\overline{C}\overline{D}}(\overline{A}+B)$；

（4）$F(W,X,Y,Z)=(W+\overline{X})(Y+\overline{Z})+(\overline{W}+X)\overline{Y}Z$。

1.15 将下列逻辑函数式化为与非-与非形式，并画出全部由与非逻辑单元组成的逻辑图。

（1）$F = AB + BC + AC$；

（2）$F = (\overline{A} + B)(A + \overline{B})C + \overline{BC}$；

（3）$F = \overline{AB\overline{C} + A\overline{B}C + \overline{A}BC}$；

（4）$F = A\overline{BC} + \overline{\overline{A\overline{B}} + \overline{\overline{A}B} + BC}$。

1.16 将 1.14 题的逻辑函数式化为或非-或非形式，并画出全部由或非逻辑单元组成的逻辑图。

1.17 利用逻辑代数的基本公式和常用公式化简下列各式。

（1）$F = A(A + \overline{B}) + BC(\overline{A} + B) + \overline{B}(A \oplus C)$；

（2）$F = \overline{(A + B)(A + C)} + \overline{A + B + C}$；

（3）$F = Y\overline{Z}(\overline{Z} + \overline{Z}X) + (\overline{X} + \overline{Z})(\overline{X}Y + \overline{X}Z)$；

（4）$F = \overline{(A + \overline{C} + D)(\overline{B} + C)(A + \overline{B} + D)(\overline{B} + C + \overline{D})}$。

1.18 用逻辑代数的基本公式和常用公式将下列逻辑函数化为最简与或形式。

（1）$Y = \overline{A}BC + A\overline{B}C + ABC$；

（2）$Y = \overline{A\overline{B}C} + \overline{A(C + D)} + BCD$；

（3）$Y = A\overline{B}D + \overline{B}CD + \overline{A + C}$；

（4）$Y = AD + BC\overline{D} + (\overline{A} + \overline{B})C$。

1.19 用卡诺图表示下列函数。

（1）$L = (A\overline{B} + AC)\overline{C} + \overline{(AB + \overline{A}\overline{C})C} + CD$；

（2）$L = \overline{C(\overline{A\overline{B}} + \overline{A}\overline{B})} + C(A + B)$。

1.20 用卡诺图化简法化简下列逻辑函数为最简与或形式。

（1）$L = A\overline{B}\overline{C} + \overline{A}\overline{B} + \overline{A}\overline{C}D + BD$；

（2）$L = ABC + ABD + A\overline{C}D + \overline{C}\overline{D} + A\overline{B}C + \overline{A}C\overline{D}$；

（3）$L(A,B,C,D) = \sum m(3,4,5,6,9,10,12,13,14,15)$；

（4）$L(A,B,C,D) = \sum m(0,1,2,5,6,7,8,9,13,14)$。

1.21 将下列具有无关项的逻辑函数化为最简的与或逻辑式。

（1）$Y(A,B,C) = \sum m(0,1,2,4) + \sum d(3,6)$；

（2）$Y(A,B,C) = \sum m(0,2,4,7) + \sum d(1,6)$；

（3）$Y(A,B,C,D) = \sum m(1,2,4,12,14) + \sum d(5,6,7,8,9,10)$；

（4）$Y(A,B,C,D) = \sum m(0,1,4,8,10,12,13) + \sum d(2,3,6,11)$。

1.22 用卡诺图将下列函数化为最简的与或逻辑式。

（1）$L = C\overline{D}(A \oplus B) + \overline{A}B\overline{C} + \overline{A}\overline{C}D$，给定约束条件为 $AB + CD = 0$。

（2）$L = (A\overline{B} + B)C\overline{D} + \overline{(A + B)(\overline{B} + C)}$，给定约束条件为 $ABC + ABD + BCD = 0$。

（3）$L(A,B,C) = \sum m(0,1,2,4)$，给定约束条件为 $m_3 + m_5 + m_6 + m_7 = 0$。

（4）$L(A,B,C,D) = \sum m(2,3,7,8,11,14)$，给定约束条件为 $m_0 + m_5 + m_{10} + m_{15} = 0$。